RL-TR-97-167
Final Technical Report
October 1997

EXPLOITING STOCHASTICITY IN SYSTEMATIC SEARCH: RESULTS ON A HIGHLY STRUCTURED DOMAIN

CALSPAN-UB RESEARCH CENTER

Carla Gomes

DTIC QUALITY INSPECTED 2

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

19980317 061

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-97-167 has been reviewed and is approved for publication.

APPROVED: *Karen M. Alguire*

KAREN M. ALGUIRE
Project Engineer

FOR THE DIRECTOR: *John A. Graniero*

JOHN A. GRANIERO, Chief Scientist
Command, Control & Communications Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2 REPORT DATE Oct 97	3. REPORT TYPE AND DATES COVERED Final Jan 96- Jan 97

4. TITLE AND SUBTITLE

EXPLOITING STOCHASTICITY IN SYSTEMATIC SEARCH:
RESULTS ON A HIGHLY STRUCTURED DOMAIN

6. AUTHOR(S)

Carla Gomes

5. FUNDING NUMBERS

C: F30602-93-D-0075, Task 30
PE: 62702F
PR: 5581
TA: 27
WU: PV

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

CALSPAN-UB RESEARCH CENTER
4455 Genesee Street
Buffalo, NY 14225

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory/C3CA
525 Brooks Road
Rome, NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-97-167

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Karen M. Alguire/C3CA/315-330-4833

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Relase/Distribution Unlimited

12b. DISTRIBUTION CODE

A

13. ABSTRACT (Maximum 200 words)

We introduce a new benchmark domain for hard computational problems that bridges the gap between purely random instances and highly structured problems. We show how to obtain interesting search problems by introducing random perturbations into a structured domain, and how such problems can be used to study the robustness of search control mechanisms. Our experiments demonstrate that the performance of search strategies designed to mimic direct constructive methods degrade surprisingly quickly in the presence of even minor perturbations. On the other hand, our experiments show that by adding a random element to a complete search procedure we can dramatically improve the performance of deterministic methods. These results apply both for the case of finding consistent models as well as for the case of proving inconsistency, complementing the well-known success of using randomness in incomplete model-find procedures. We pushed the idea of exploiting stochasticity one step further by combining algorithms that exhibit competing behavior into portfolios. Our results show that a portfolio approach can have a dramatic impact in terms of the overall performance, in comparison to the performance of each of its component algorithms. An interesting special case is when the best strategy consists of combining copies of the same algorithm. This portfolio is analogous to the practice of ``restarts'' for stochastic procedures, where the same algorithm is run repeatedly with different seeds, a common practice in the theorem-proving community. Combining copies of the same algorithm into a portfolio might be a good strategy, especially in the cases that one algorithm dominates for a given class of problems.

14. SUBJECT TERMS

Constraint Satisfaction, algorithm portfolios, quasigroup completion problem, search control strategies

15. NUMBER OF PAGES
60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Exploiting Stochasticity in Systematic Search: Results on a Highly Structured Domain

Carla P. Gomes
Rome Laboratory*
gomes@ai.rl.af.mil

Abstract

We introduce a new benchmark domain for hard computational problems that bridges the gap between purely random instances and highly structured problems. We show how to obtain interesting search problems by introducing random perturbations into a structured domain, and how such problems can be used to study the robustness of search control mechanisms. Our experiments demonstrate that the performance of search strategies designed to mimic direct constructive methods degrade surprisingly quickly in the presence of even minor perturbations. On the other hand, our experiments show that by adding a random element to a complete search procedure we can dramatically improve the performance of deterministic methods. These results apply both for the case of finding consistent models as well as for the case of proving inconsistency, complementing the well-known success of using randomness in incomplete model-find procedures. We pushed the idea of exploiting stochasticity one step further by combining algorithms that exhibit competing behavior into *portfolios*. Our results show that a portfolio approach can have a dramatic impact in terms of the overall performance, in comparison to the performance of each of its component algorithms. An interesting special case is when the best strategy consists of combining copies of the same algorithm. This *portfolio* is analogous to the practice of "restarts" for stochastic procedures, where the same algorithm is run repeatedly with different seeds, a common practice in the theorem-proving community. Combining copies of the same algorithm into a *portfolio* might be a good strategy, especially in the cases that one algorithm *dominates* for a given class of problems.

---

*Carla P. Gomes works for Rome Laboratory as a Research Associate.

1 Introduction

A lot of progress has been made in the understanding of the computational complexity of hard problems through experimental work. Due to the difficulty in gathering realistic data, researchers heavily resort to random instance distributions. Using random instance distributions, hard search problems have been identified. Such instances have pushed the development of new search methods, both in terms of systematic and stochastic procedures (Hogg et al. 1996).

The study of highly structured problems, such as those from various finite algebra domains, has also driven the development of search procedures. For example, the question of the existence and non-existence of certain discrete structures with intricate mathematical properties gives rise to some of the most challenging search problems (Lam et al. 1989; Fujita et al. 1993; McCune 1996).

An important question is to what extent real-world search and reasoning tasks are represented by such problems. It seems clear that random instances lack certain structure that is often present in realistic problems. On the other hand, the highly structured mathematical problems contain too much structure from the perspective of realistic applications.

In this report we propose a new benchmark domain that bridges the gap between the purely random instances and the highly structured problems. We consider structured problems from the area of combinatorics. Our goal is to study the properties of such structured domains in the presence of perturbations to their structure.

As our starting point, we selected the quasigroup, a basic discrete structure that can be characterized by a set of simple properties. As we will see, finding basic quasigroups is a relatively easy problem, for which direct constructions are known. However, the nature of the problem changes dramatically if we impose additional constraints that are locally consistent but not necessarily globally consistent. In particular, we *perturb* the structure of the quasigroup by requiring that it satisfies an incomplete initial pattern. Even though this initial pattern is consistent with the properties of quasigroups, there are no guarantees that a complete quasigroup can be derived

from it.

The quasigroup completion problem enables us to study the impact of perturbations on the complexity of the underlying well-structured problem. Using the spectrum of difficulty of the quasigroup completion problem, we study the performance of various forms of search control.

Our experiments show that the performance of search heuristics that mimic direct constructive methods for quasigroups degrade rapidly in the presence of even the smallest possible perturbations. Somewhat surprisingly, this finding suggests that it can be counterproductive to design search control strategies that specifically replicate constructive search methods. Specialized search control has of course been shown to be quite effective in certain domains. Our results merely question the robustness of such sophisticated search methods in the presence of perturbations. Apparently, even very small perturbations can "throw off" the specialized search control, making it inferior to a more general method. These results are consistent with the empirical finding that simple generic search methods often outperform more sophisticated ones when applied to a range of problem instances.

On the other hand, our experiments showed that more generic heuristics, which do not perform so well on the original problem, degrade much more gracefully in the presence of perturbations, and quickly outperform the search method that replicated a constructive method. To some extent, for general search procedures, small perturbations actually facilitate the search for solutions, $i.e.$, the introduction of random perturbations to the original problem *helps* the more generic heuristics, especially for lower levels of perturbations. This result led us to conjecture that adding a stochastic element to deterministic procedures would improve their performance.

In recent years we have seen the development of stochastic search methods. However, the main emphasis of stochastic procedures has been in terms of local search approaches. Local search procedures have proven very powerful. For example, local search methods are among the best known procedures for solving some hard classes of satisfiability problems. Nevertheless, due to their inherent incomplete nature, local search methods are limited *a priori*, since they are unsuitable for proving inconsistency.

In this report we study the use of stochastic strategies in conjunction with

complete search methods, using the quasigroup completion problem as our benchmark domain. We start by examining the effect of adding a randomized element into complete search methods *per se*. The resulting procedures retain the completeness property, *i.e.*, they always find a solution or prove that it does not exist, but their run time profiles vary from instance to instance. They are referred to as Las Vegas style algorithms.

Our results show that by adding a random element to a complete search procedure we can dramatically improve the performance of the deterministic methods, for a given level of *risk*. These results apply both for the case of finding consistent models as well as for the case of proving inconsistency, complementing the well-known success of using randomness in incomplete model-find procedures.

A third aspect that we examine in this report is the combination of different stochastic algorithms. Various approaches have been developed to combine different algorithms to take into account the computational resource constraints and to optimize the overall performance. This research has led to the development of anytime algorithms (Dean and Boddy 1988), decision theoretic metareasoning and related approaches (Horvitz and Zilberstein 1996; Russell and Norvig 1995), and algorithm portfolio design (Huberman *et al.* 1997). However, despite the numerous results obtained in these areas, so far they have not been exploited much by the traditional communities that study hard computational problems, such as operations research (OR), constraint satisfaction (CSP), theorem proving, and the experimental algorithms community.

In order to bridge this apparent gap, we study the possibility of combining algorithms in the context of the recent results concerning the inherent complexity of computationally hard search and reasoning problems. We provide a rigorous empirical study of the performance profiles of several of the state-of-the-art search methods on a distribution of hard search problems. We use the quasigroup completion problem as our benchmark domain.

We discuss the conditions under which the composition of different algorithms into a *portfolio* can have a dramatic impact in terms of the overall performance, in comparison to the performance of each of its component algorithms. As a particular case, we discuss the strategy of creating a *portfolio*

4

consisting of different copies of the same algorithm. This *portfolio* is analogous to the practice of "restarts" for stochastic procedures, where the same algorithm is run repeatedly with different seeds. Quite often this strategy of combining copies of the same algorithm into a *portfolio* has a tremendous pay-off, especially in the cases where one algorithm *dominates* for a given class of problems.

This report is structured as follows. In the next section, we introduce quasigroups and define the quasigroup completion problem. We also discuss the theoretical complexity of the problem. In section 3 we present empirical results on the quasigroup completion problem. Section 4 contains the evaluation of deterministic search strategies and their scaling properties. In section 5 we introduce several complete stochastic search methods and, in section 6, we give their performance distribution profiles on the quasigroup completion problem. In section 7 we design and evaluate various algorithm portfolios. Finally, in section 8, we summarize our results and discuss future directions.

This work has been done in collaboration with Bart Selman (Gomes and Selman 1997a and 1997b).

2 The Quasigroup Completion Problem

A quasigroup is an ordered pair $(Q, \cdot)$, where Q is a set and $(\cdot)$ is a binary operation on Q, such that the equations $a \cdot x = b$ and $y \cdot a = b$ have a unique solution for every pair of elements a, b in Q. The *order* N of the quasigroup is the cardinality of the set Q. A good way to understand the structure of a quasigroup is to consider its N by N multiplication table as defined by its binary operation. (For each pair of elements x and y, the table gives the result of $x \cdot y$.) The constraints on a quasigroup are such that its multiplication table defines a *Latin square*. This means that in each row of the table, each element of the set Q occurs exactly once; similarly, in each column, each element occurs exactly once (Denes and Keedwell 1974).

An *incomplete* or *partial latin square* P is a partially filled N by N table such that no symbol occurs twice in a row or a column. The *Quasigroup Completion Problem* is the problem of determining whether the remaining entries of the table can be filled in such a way that we obtain a complete latin square, that is, a full multiplication table of a quasigroup. We view the pre-assigned values of the latin square as a *perturbation* to the original problem of finding an arbitrary latin square. As we will discuss below, there are direct constructive methods for generating a latin square of any order. However, the situation is quite different for completing a partial latin square.

Evans (1960) conjectured that every N by N partial latin square with at most $(N - 1)$ cells occupied can be completed to a latin square of order N. This is known as the *Evans conjecture*. Despite the fact that the problem received much attention, and many partial solutions were published, it took until 1981 for the conjecture to be proved correct (Smetaniuk 1981).

Andersen and Hilton (1983), through independent work, proved Evans conjecture with stronger results. They give a complete characterization of those partial latin squares of order N with N non-empty cells that *cannot* be completed to a full latin square. However, it appears unlikely that one can characterize non-completable partial latin squares with an *arbitrary* number of pre-assigned elements. This is because the completion problem was shown to be NP-complete by Colbourn (1983, 1984). Of course, this makes the problem computationally interesting from the perspective of search and constraint satisfaction.

An interesting application area of latin squares is the design of statistical experiments. The purpose of latin squares is to eliminate the effect of

certain systematic dependency among the data (Denes and Keedwell 1974). Another interesting application is scheduling and timetabling. For example, latin squares are useful in determining intricate schedules involving pairwise meetings among the members of a group (Anderson 1985). The natural perturbation of this problem is the problem of completing a schedule given a set of pre-assigned meetings.

Connected to our work is the work in the area of automated theorem proving, in particular the work on quasigroups and finite algebra in general (Lam et al. 1989; Fujita et al. 1993; Stickel 1994; McCune 1996). One interesting question is to what extent special search heuristics that can guide the search to find general unrestricted quasigroups are also of use in finding special quasigroups with interesting mathematical properties. In addition, the notion of completing partial solutions may be useful in exploring the total space of solutions. As discussed below, partial instantiations of the quasigroups can actually guide the search. The completion problem also provides some insights into the density of solutions. For example, easy completion of partial structures does suggest a high density of solutions.

3 Computational Results for the Quasigroup Completion Problem

We now consider the practical computational difficulty of the quasigroup completion problem. There is a natural formulation of the problem as a Constraint Satisfaction Problem. We have one variable for each of the N^2 entries of the multiplication table of the quasigroup, and we use constraints to capture the requirement of having no repeated values in any row or column. All variables have the same domain, namely the set of elements Q of the quasigroup. Pre-assigned values are captured by fixing the value of some of the variables.

A natural question to consider is how the difficulty of the quasigroup completion problem depends on the number of pre-assigned values. Figure 1 gives the median number of backtracks needed to find a completion or to show none exists. Along the horizontal axis we give the fraction of pre-assigned values (out of a total of N^2 values, where N is the order of the quasigroup). From the figure, we observe that the costs peak roughly around the same ratio (approximately 42% pre-assignment) for different values of N. Figure 2 uses a log-scale plot to better show the scaling behavior.

What we observe here is a clear phase-transition in the problem domain. Figure 3 further confirms this. It plots the fraction of unsolvable cases as a function of the fraction of pre-assigned elements in the quasigroup. We see that the transition — from almost all instances solvable to almost all unsolvable — occurs around the same ratio as the peak in the computational difficulty. (Each data point is generated using 1,000 problem instances. The pre-assigned values were randomly generated. We used the specialized control heuristic described below.)

The more detailed median cost curves in Figure 2 show an interesting asymmetry. The right-hand side of the median cost peak is very steep, which shows that instances become easily solvable or proved unsolvable. The slope on the left-hand side is much less abrupt. Similar asymmetries have been observed in, for example, the work on random Boolean satisfiability (SAT) problems (Mitchell *et al.* 1992; Crawford and Auton 1993; Kirkpatrick and Selman 1994). However, the slope in those curves is much steeper on the left-hand side, *i.e.*, the satisfiable area. The slope on the right-hand side is much less steep because proving unsatisfiability of random Boolean expres-

sions requires large search trees (Chvatal and Szemeredi 1988). It's quite interesting to see the opposite phenomenon here. The difference may be due to the fact that we start from a highly structured problem with the randomness being introduced only as a perturbation. In the random SAT problems all the constraints (Boolean disjunctions) are randomly generated.

This is in contrast with our approach, where we start with a highly regular original set of constraints, defining the quasigroup structure. We then perturb this structure by randomly pre-assigning a set of values. Therefore, our instances combine a relatively high degree of structure with an element of irregularity or uncertainty due to the pre-assigned values. The fact that we again observe a clear phase transition phenomenon is evidence for the practical relevance of the phase transition work across a wide range of constraint satisfaction problems. See, for example, Cheeseman *et al.* (1991), Crawford and Auton (1993), Mitchell et al. (1992), Kirkpatrick and Selman (1994), and Smith and Dyer (1996). Hogg *et al.* (1996) contains a collection of recent papers in the area.

Some important related work on hard problem instances involving some underlying structure is that of Gent and Walsh (1995) and Zhang and Korf (1996). Both teams use the structure of combinatorial optimization problems, such as the Traveling Salesman Problem and real-world Timetabling problems. By varying the constraint density of their problem instances they obtain varying degrees of difficulty. One important difference is that in our approach we start with an initial problem structure for which a direct construction is known. As we will discuss below, this allows us to consider search control mechanisms that mimic the constructive methods and study their robustness under perturbations. We also believe that our domain fits more naturally with reasoning style problems ("satisfaction problems") as opposed to optimization style problems.

In the next section we discuss how our problem instances enable us to evaluate the robustness of different search strategies. We will show that the relative performance of search strategies varies, depending on the location of the problem instances with respect to the phase transition. This suggests ways of chosing search control depending on the statistical properties of the problems under consideration.

Finally, let us briefly contrast the type of behavior observed for our in-

stances with another, less constrained structured problem, namely the now "infamous" N-queens problem. This problem used to be a fairly popular benchmark within the CSP community but recently the problem has been shown to be surprisingly easy, at least for certain stochastic methods. To investigate the structure of the solution space of this problem we consider the completion problem for the N-queens. The idea is to provide an initial partial placement of non-attacking queens on the board and consider the question of whether this initial pattern can be completed to a placement of N non-attacking queens. Figure 4 shows the phase transition for this problem. From the figure, it is clear that the phase transition in this case is not confined to a single region, but rather the transition shifts to the right. In some sense, we're dealing with a "vanishing" phase transition. More accurately, in the limit, for large values of N, the transition shifts to 1. This means that we can randomly pre-place up to $(N-1)$ queens (which does not appear to be difficult), and still be able to successfully complete the pattern. This provides a nice intuitive explanation for the success of the stochastic procedures on this problem. Such procedures can solve the N-Queens problem with, for example, one million queens in under a minute on a Sparc workstation (Minton et al. 1991; Gu 1989). It would be interesting to find a provable polynomial procedure for the N-Queens completion task or to show the problem is NP-complete. Note that there are known constructive methods for dealing with the case of zero preplacements. See, for example, Yaglom and Yaglom (1964), Bruen and Dixon (1974), Falkowski and Schmitz (1986), and Erbas *et al.* (1992).
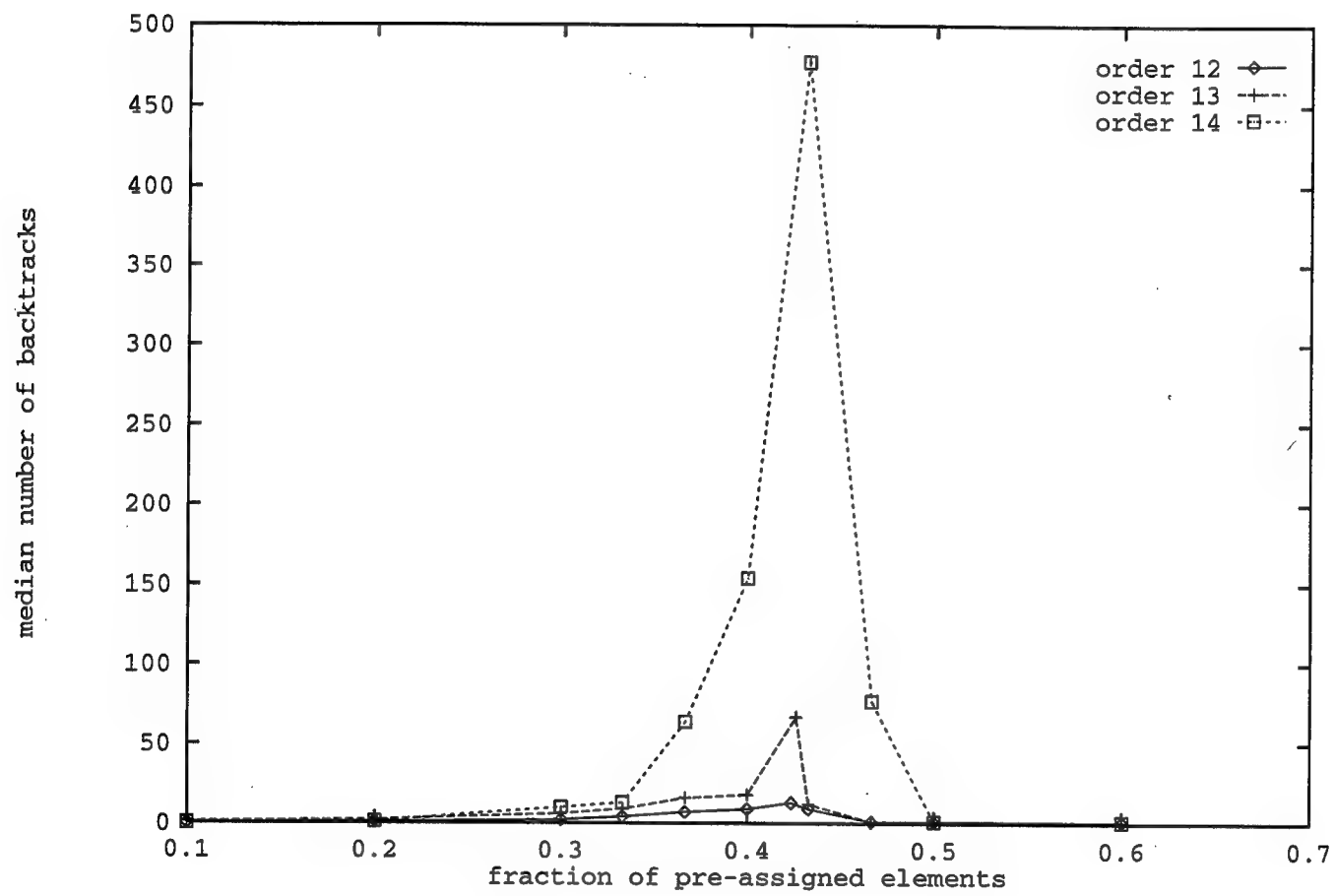
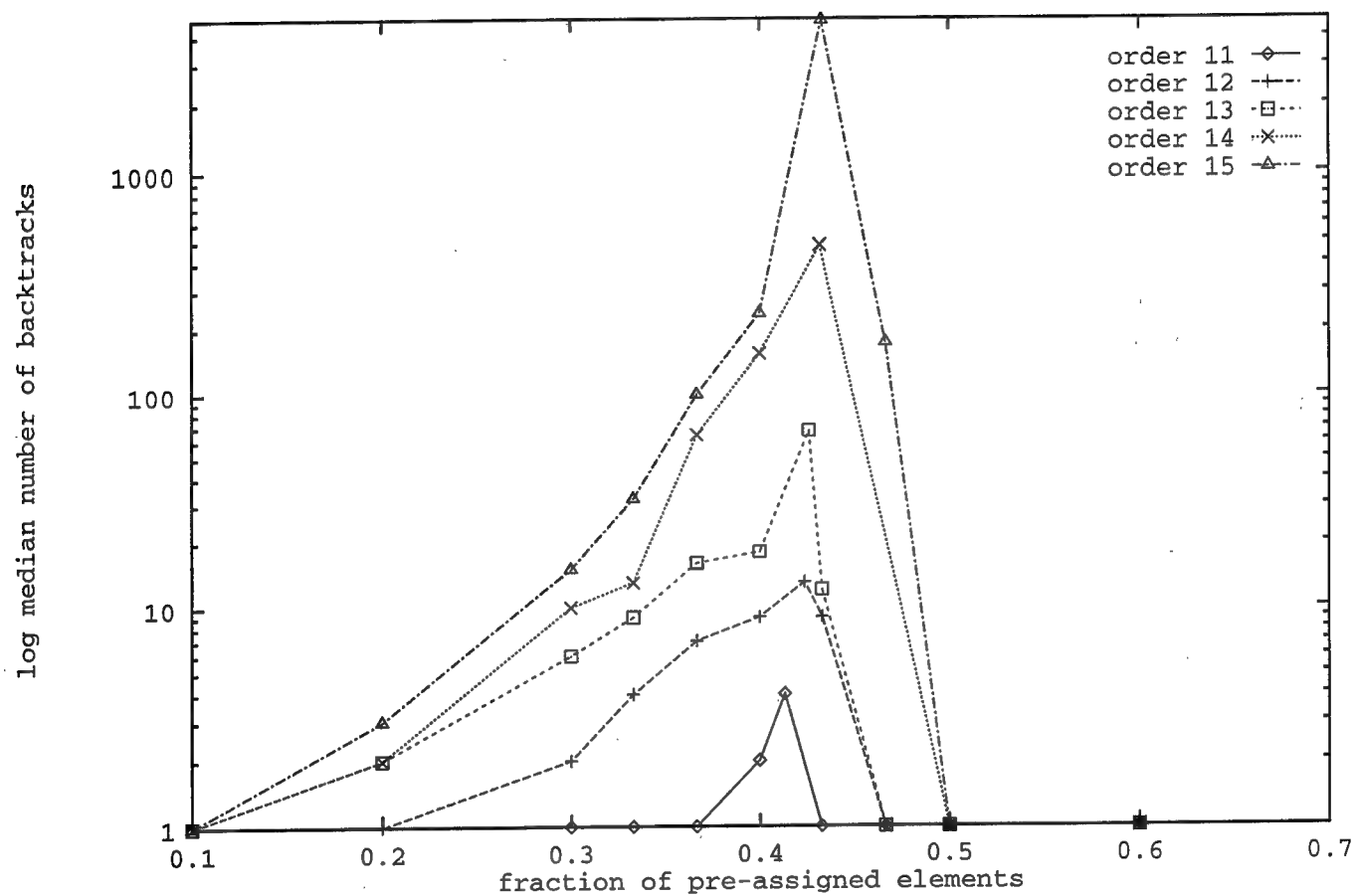Figure 1: The Complexity of Quasigroup Completion

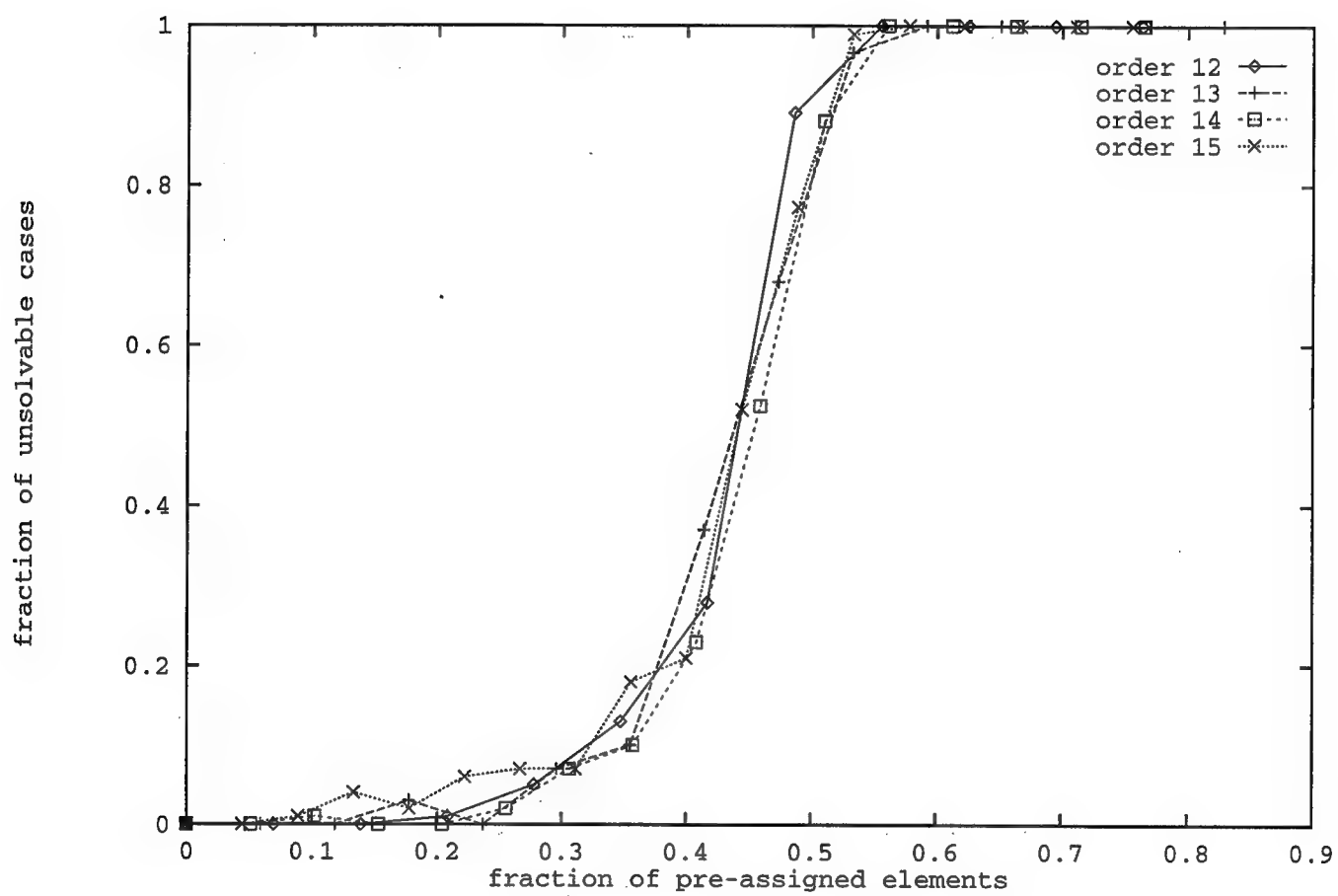Figure 2: The Complexity of Quasigroup Completion (Log Scale)

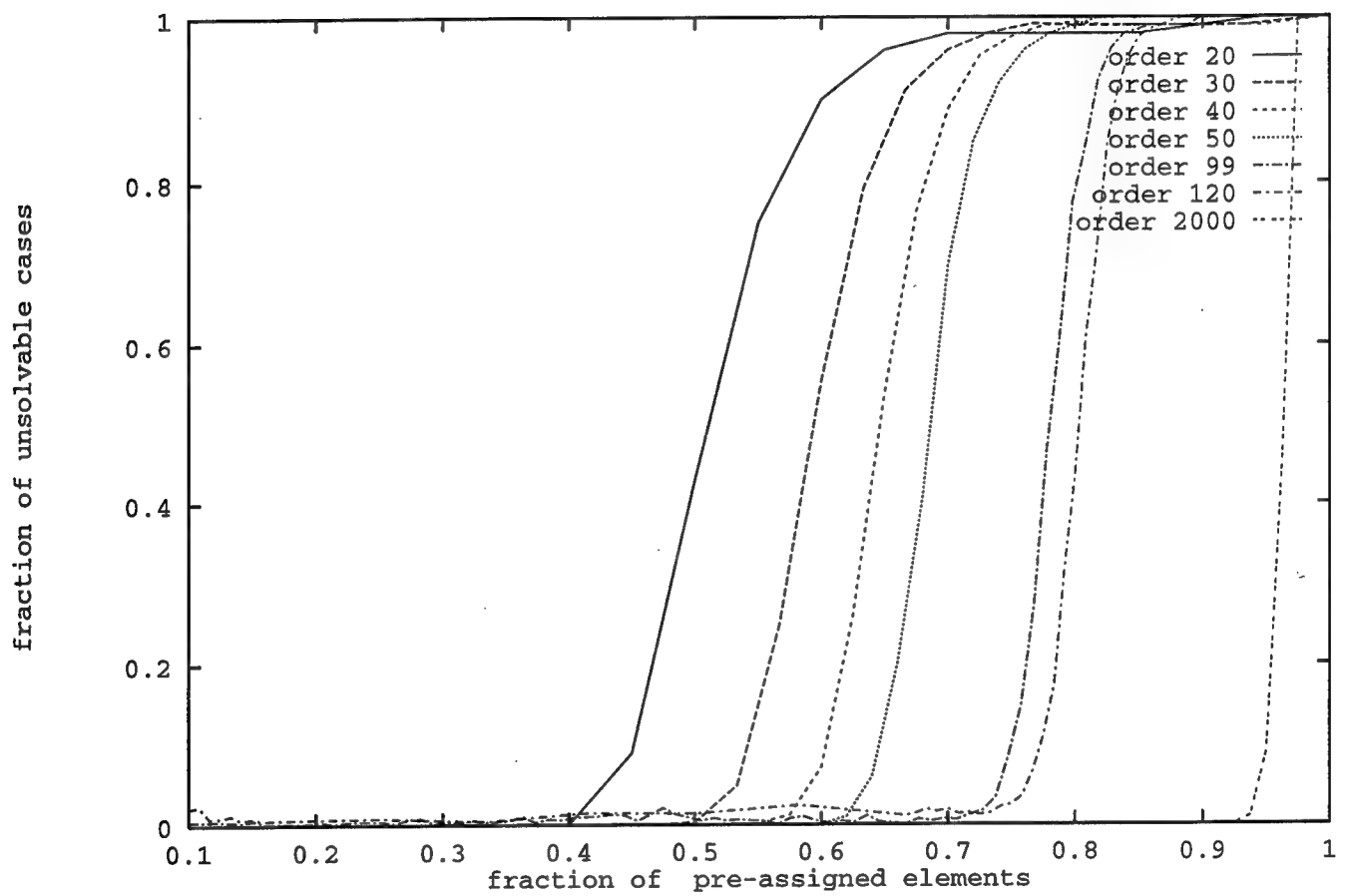Figure 3: Phase Transition for the Completion Problem

Figure 4: A Shifting Phase Transition

14

4 A Comparison of Deterministic Search Heuristics

There are several direct methods for generating basic quasigroups of any order N. For example, consider placing the elements of the quasigroup in the first row of the multiplication table in some arbitrary order. Now, we can generate the next row, by simply shifting the elements from the first row, one place to the right (the rightmost element wraps around and moves to the leftmost position of the second row). We repeat this process for the third row by shifting the pattern of the second row another cell to the right. The remaining rows are filled in a similar manner. By this construction, it is clear that there will be no repeated elements in any row, nor in any column. We therefore end up with a latin square of order N, which again defines the multiplication table of a quasigroup.

We can mimic this constructive method using a backtrack procedure with appropriate search control. In particular, we can add a tie-breaking criterion to the generic First-Fail heuristic. In the First-Fail heuristic, one selects as the next branching variable the one with smallest remaining domain. We add the following tie-breaking criterion: select from among the variables with the smallest current domains, the variable with the smallest minimal value in its current domain. (We can assume some arbitrary fixed order on the quasigroup elements.)

We encoded this problem in C++ using ILOG SOLVER, a powerful C++ constraint programming library (Puget 1994). ILOG provides a backtracking mechanism that allows us to keep track of variables and their domains, while maintaining arc-consistency (van Hentenryck *et al.* 1992).

Our experiments below show that the First-Fail heuristic with the tie-breaking as defined above generates quasigroups of any order with zero backtracks. So, the tie-breaking rule effectively mimics a constructive method for the quasigroup. The key question is how robust such a heuristic is in the presence of perturbations to the structure.

In Figure 5 we compare the performance of this specialized heuristic with the more general First-Fail strategy on the quasigroup completion problem with no pre-assigned values. (So, the task is to generate any quasigroup of order N.) The figure uses a logarithmic scale. In order to show the specialized heuristic on instances with 0 pre-assigned values using a logarithmic scale,

we added "2" to its median number of backtracks. The figure clearly shows that the specialized control indeed does not create any backtracks. However, the number of backtracks for the general heuristic increases sharply with increasing N, reaching our cutoff of $100,000$ when N is 32.

We now consider the change in behavior when we slightly perturb the quasigroup structure by pre-assigning 5% of the elements of its multiplication table. See Figure 6. The figure shows the performance of our specialized heuristic degrades dramatically. In fact, the general First-Fail heuristic without tie breaking actually scales substantially better than our customized control. (Note the logarithmic scale.) In other words, the specialized control heuristic is surprising "fragile" in the presence of small perturbations.

Figure 7 further confirms the difference in performance of the specialized and the general heuristic. This time we consider problems in the hardest area of the phase transition (about 40% pre-assignment). We see that the specialized tie-breaking rule performs consistently worse on the hardest quasigroup completion problems, near the phase transition. Notice that even on a log scale the gap between the heuristics actually widens for larger N. (Our search cutoff actually hides some of this difference.) So, except for completely unperturbed versions of our search problem, the First-Fail heuristic with tie-breaking is consistently worse than the First-Fail heuristics by itself. This despite the fact that the tie-breaking rule mimics a constructive method for the quasigroup structure in the unperturbed version.

Figure 8 illustrates in more detail the performance of the general heuristic. With a small number of pre-assigned values (1%), the heuristic performs better than with no pre-assigned values at all. However, its performance deteriorates again in the hard area of the phase transition. Apparently, the small number of randomly pre-assigned values helps the heuristic in finding a completion of the quasigroup. This is consistent with the fact that the specialized heuristic actually performs worse. The tie breaking strategy in this heuristic makes it more deterministic, which appears to hurt its performance on the completion task.

To summarize, we observed that the performance of a specialized search control — designed to mimic a constructive method to generate quasigroups — degrades rapidly in the presence of small perturbations. A more general heuristic appears more robust. In fact, we saw how small perturbations can actually improve the performance of such a heuristic. We believe that randomness versus determinism plays a key role in these phenomena. These

16

findings suggest care should be taken in the use of tailored heuristics: their performance can degrade dramatically in the presence of minor perturbations. In the presence of such perturbations, less sophisticated, more generic heuristics may very well have better scaling properties. This is an interesting issue for further exploration. In the next section we study the effect of directly introducing a random element into a complete search method.

Closely connected to our work is the study of search control and various constraint processing techniques (Ginsberg, Korf, van Beek, Freuder, Dechter, Prosser, Smith, Freuder), and the work on selecting appropriate search heuristics (see eg. Minton). We hope that our approach will stimulate more research into the robustness of the various methods. The idea of completing partial solutions is, of course, applicable in many CSP domains. Therefore, by asserting partial solutions, that are at least locally consistent, one can study whether special search techniques degrade gracefully.
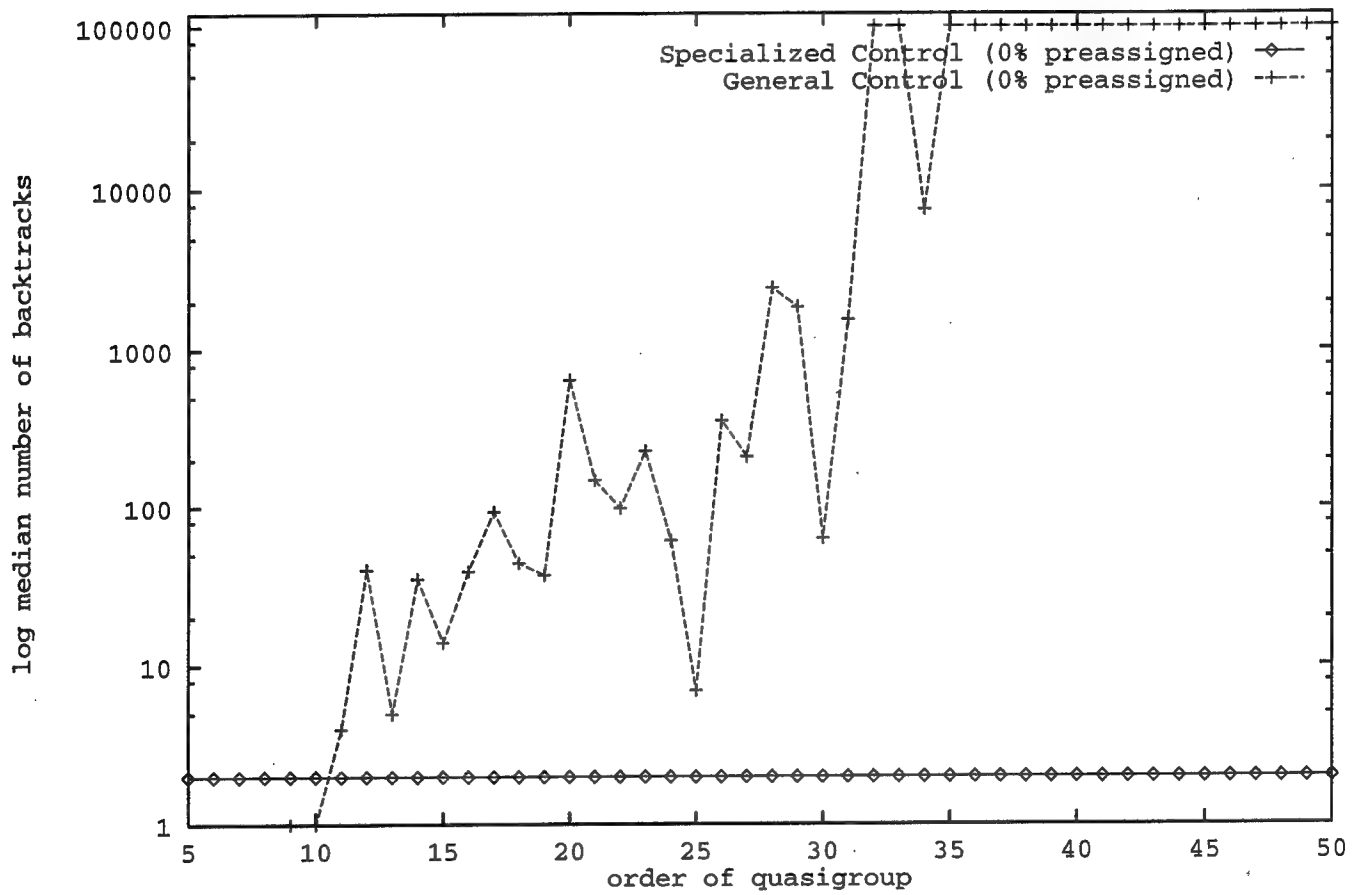
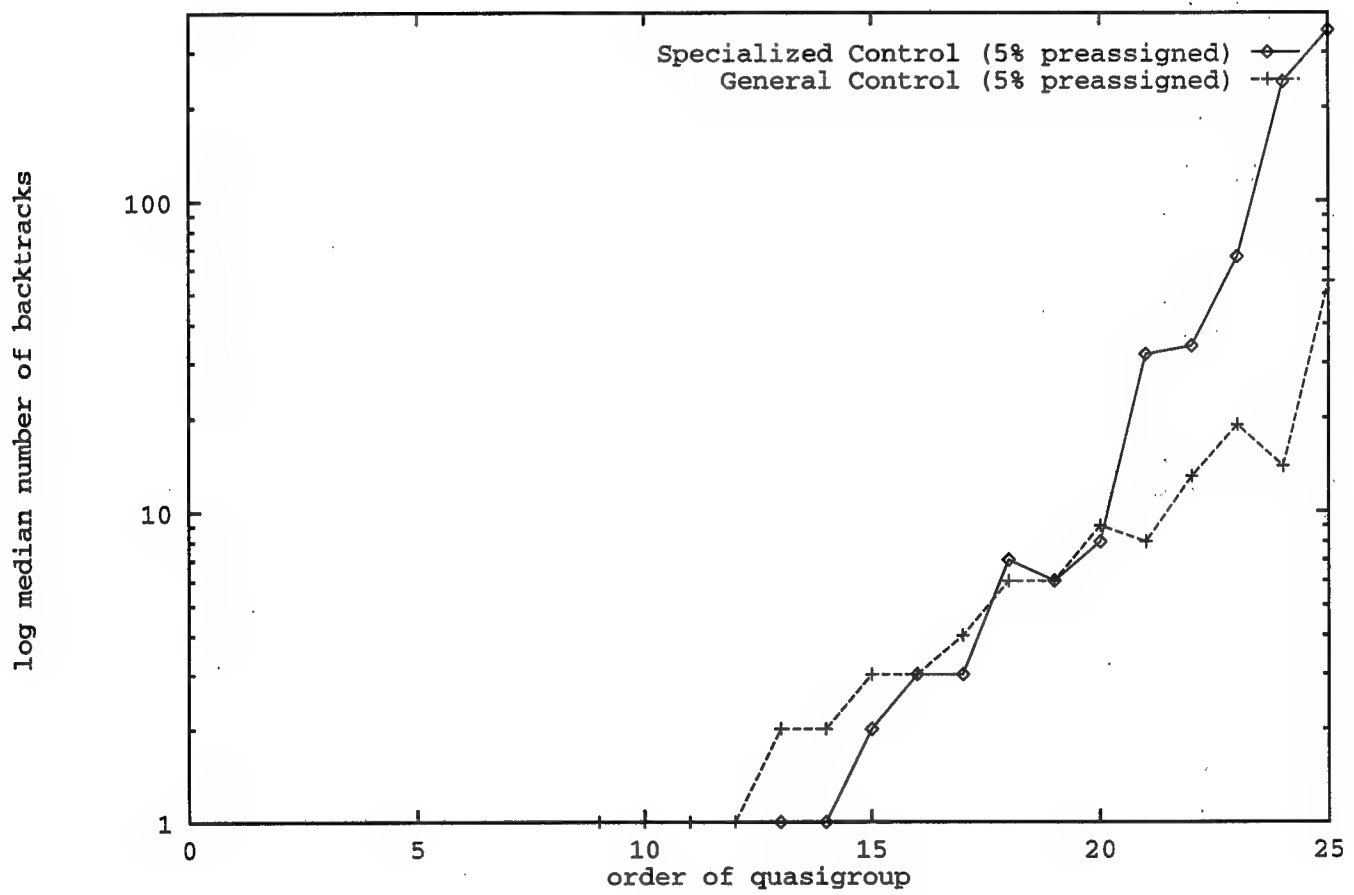Figure 5: General vs. Specialized Search Control (0% Pre-assignment).

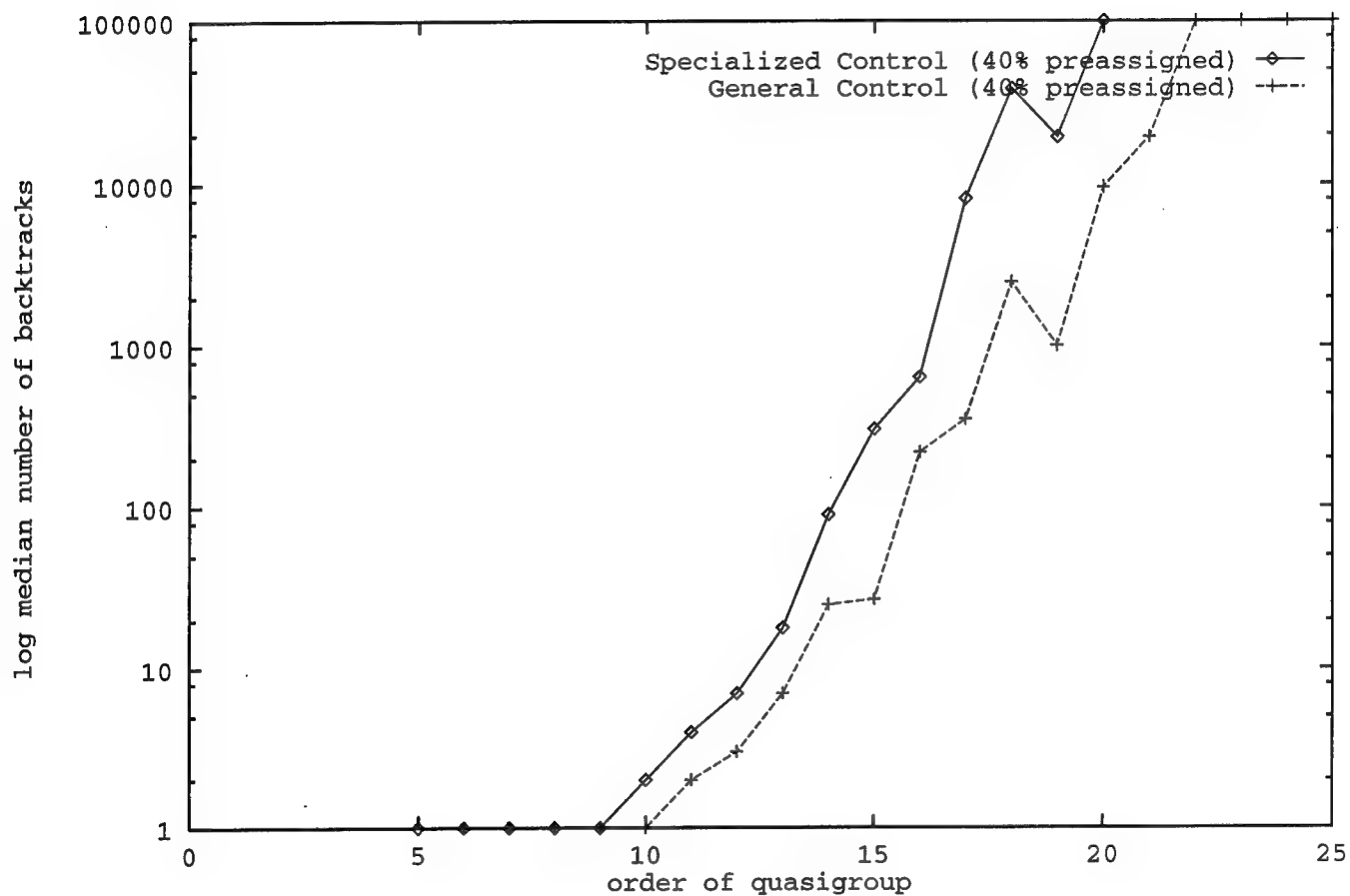Figure 6: General vs. Specialized Search Control (5% Pre-assignment).

Figure 7: General vs. Specialized Search Control with 40% Perturbation
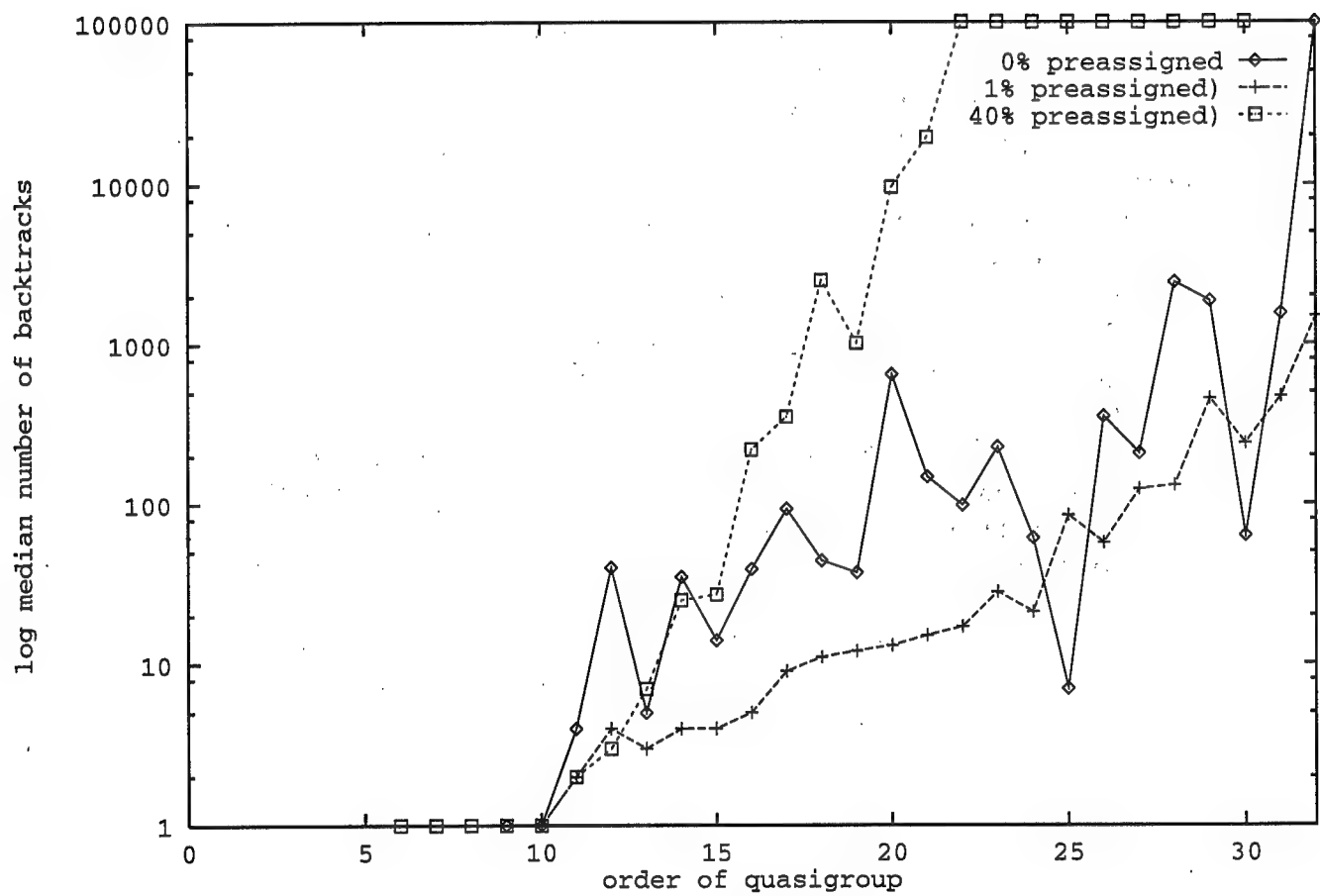
Figure 8: General Search Control. Effect of Pre-assignment.

5 Exploiting Stochasticity

In the previous section we considered the performance of the generic First-Fail search heuristic and compared it to a more specialized strategy on the quasigroup completion problem. Our specialized control generates quasigroups without backtracking for the case with no pre-assigned values. In some sense, the method mimics a constructive method for generating quasigroups. However, as we showed, the strategy rapidly deteriorates in the presence of even very minor perturbations (only 5% pre-assigned values). On the other hand, we found that such small perturbations actually substantially *improve* the performance of the general First-Fail heuristic.

The pre-assigned values add a stochastic element to the original set of the quasigroup problem constraints. This led us to conjecture that adding a stochastic element to the First-Fail search procedure itself would improve its performance. Below we describe the experiments that we performed to explore this conjecture. We considered various ways of introducing a stochastic element to the First-Fail search procedure: both in terms of variable selection and value selection.

Our base strategy is the First-Fail principle with systematic variable selection (i.e., breaking ties in lexicographic order) and systematic value selection (i.e., lexicographically first from remaining domain). We refer to this strategy as "SS". Next, we introduce to this basic strategy a random tie breaking rule for the variable selection, while maintaining systematic value selection ("RS"). A third strategy is to keep the systematicity for the variable selection, but to select values at random from the remaining domain values ("SR"). As a final, fourth strategy, we introduce random tie-breaking for the variable selection, as well as, random value selection ("RR").

We should stress that we maintain completeness in our approach. As mentioned before, we encoded the different search strategies in ILOG SOLVER (Puget 1994). ILOG's underlying backtracking mechanism allows us to keep track of variables and their domains, while maintaining arc-consistency (van Hentenryck *et al.* 1992). This approach should be contrasted with various random "probe" strategies, in which one uses random variable selection in going down a branch of the search tree but no record is kept of which parts of the search space have been visited. An implementation of such probing strategy can be very efficient but unfortunately one loses completeness. (For a debate on some of the tradeoffs involved, see Freuder *et al.* 1995). Closely

connected to our work is the study of search control and various constraint processing techniques (Dechter 1991, Freuder *et al.* 1995, Ginsberg and Geddis 1991, Kondrak and van Beek 1995), and the work on selecting appropriate search heuristics (see eg. Minton 1996).

Figure 9 shows the performance of our various strategies on the problem of finding quasigroups with no values pre-assigned. From the four strategies, we see that the ones that have a stochastic element clearly outperform the completely deterministic procedure ("SS"). This confirms our initial intuition that an element of randomness introduced directly into the First-Fail search strategy itself would improve its performance on the structured quasigroup search problem.

From the other three stochastic strategies, "SR" appears to perform somewhat better than the other two but this difference may not be qualitatively significant.

In Figure 10 we show the performance of our strategies on the quasigroup completion problem with 5% pre-assigned values. Again, the stochastic element improves upon the base strategy, but the effect is somewhat less dramatic than in the case of no pre-assigned values. The main reason for the diminished difference in the strategies appears to be due to the fact that randomness is now also introduced into the problem via the randomly pre-assigned quasigroup values. In other words, perturbations introduced into the quasigroup constraints (through the pre-assigned values) help the deterministic First-Fail heuristic in a similar way as the stochastic element introduced directly into the search method itself, as in our "RS", "SR", and "RR" strategies.

Figure 11 shows the behavior of our procedures in the hardest region of our search space, namely at the phase transition point. In this case, the random element of the underlying problem instances overshadows the positive effect of the stochastic element in our search strategies. However, our stochastic strategies still dominate the deterministic one for dealing with some of the hardest instances in this area, namely to show inconsistency. This is not directly apparent from Figure 11, because it only gives the median cost values. We will return to this issue in the next section.

To summarize, our empirical results confirm that stochasticity can improve upon the purely deterministic strategy, both for showing consistency and inconsistency. The resulting procedures (First-Fail with a stochastic element) are more robust then either First-Fail by itself or First-Fail with lex-

icographic tie-breaking , since it performs competitively over the full range
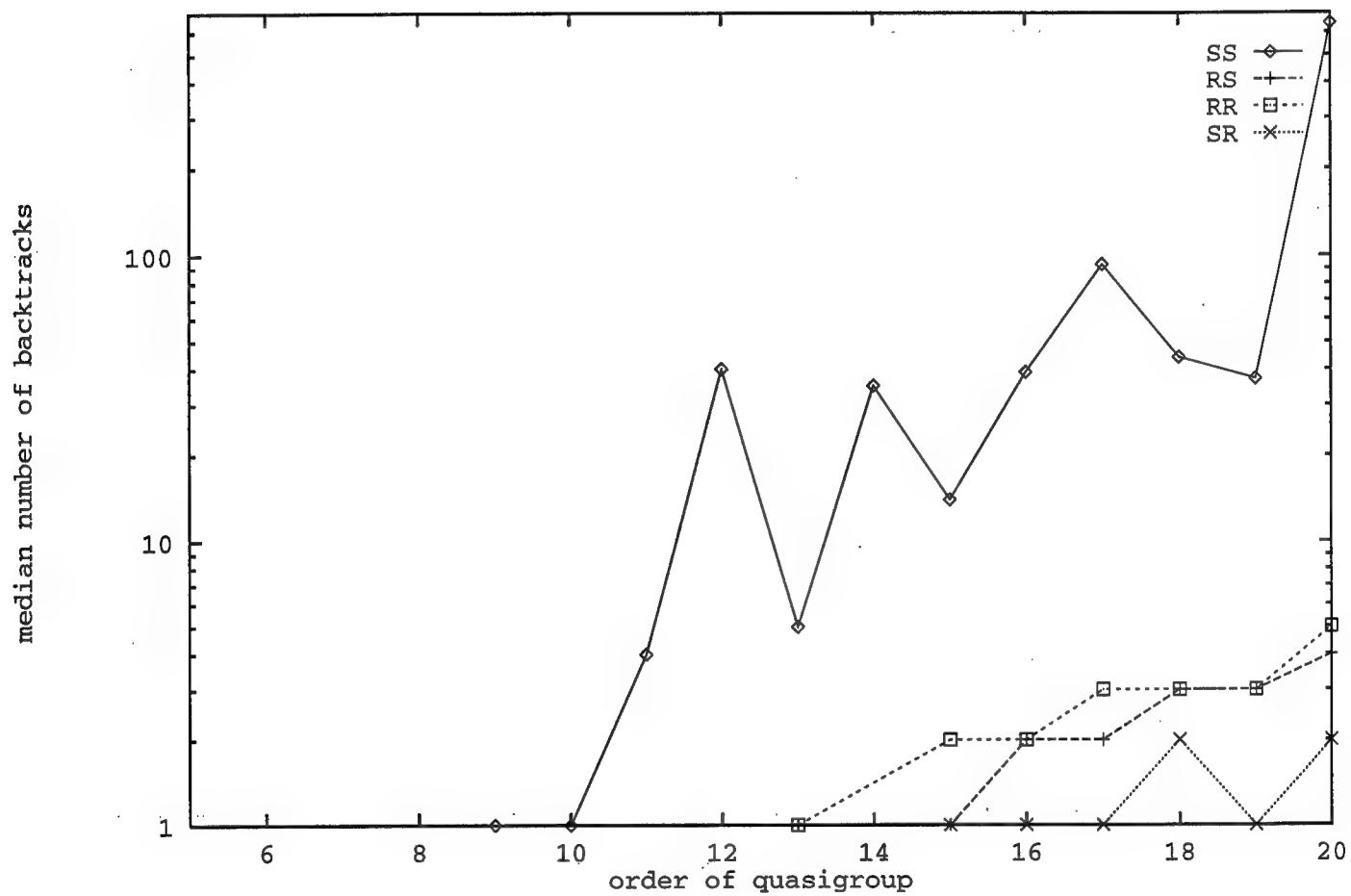of pre-assigned values (from 0% and up).

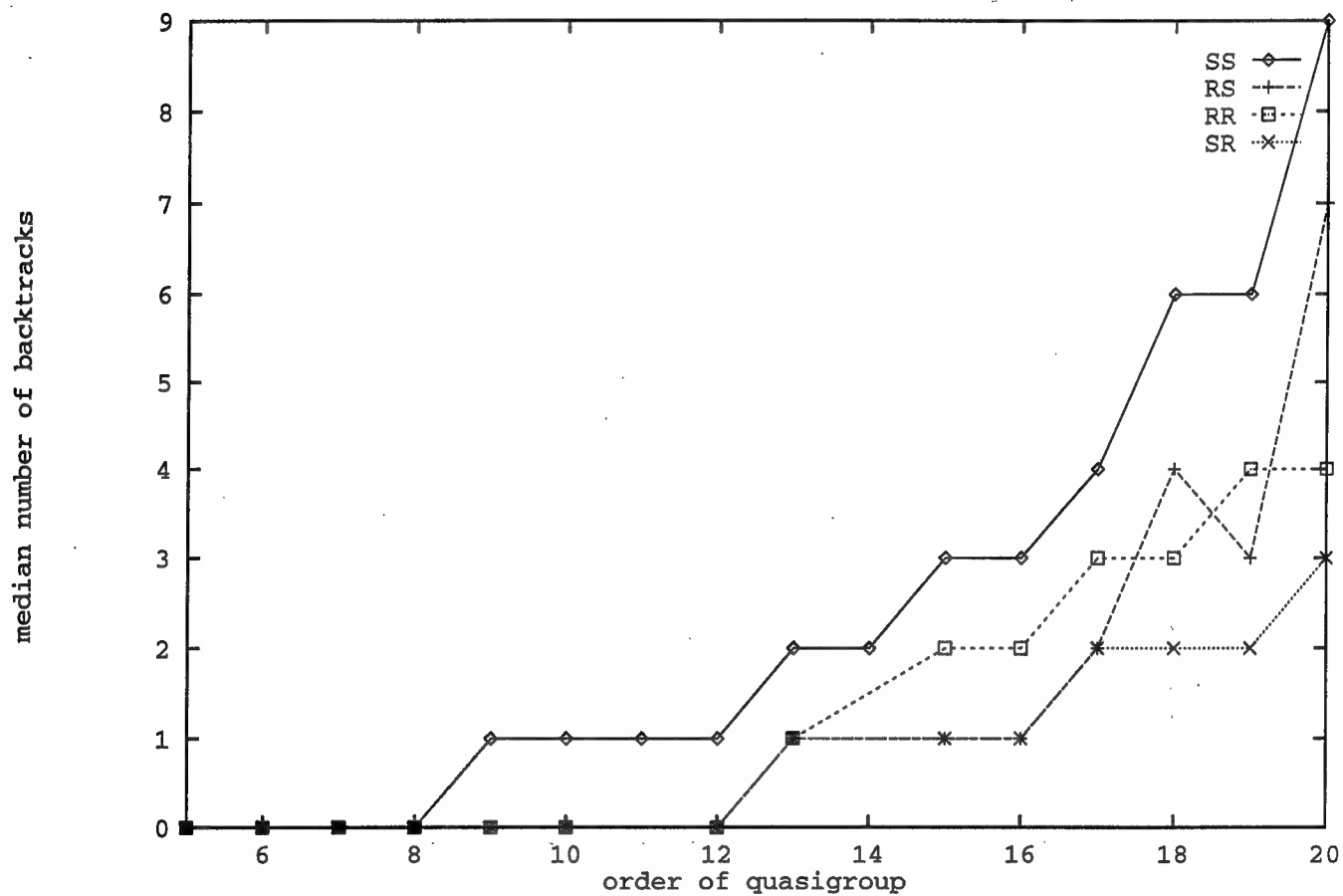Figure 9: Finding quasigroups (no pre-assigned values).

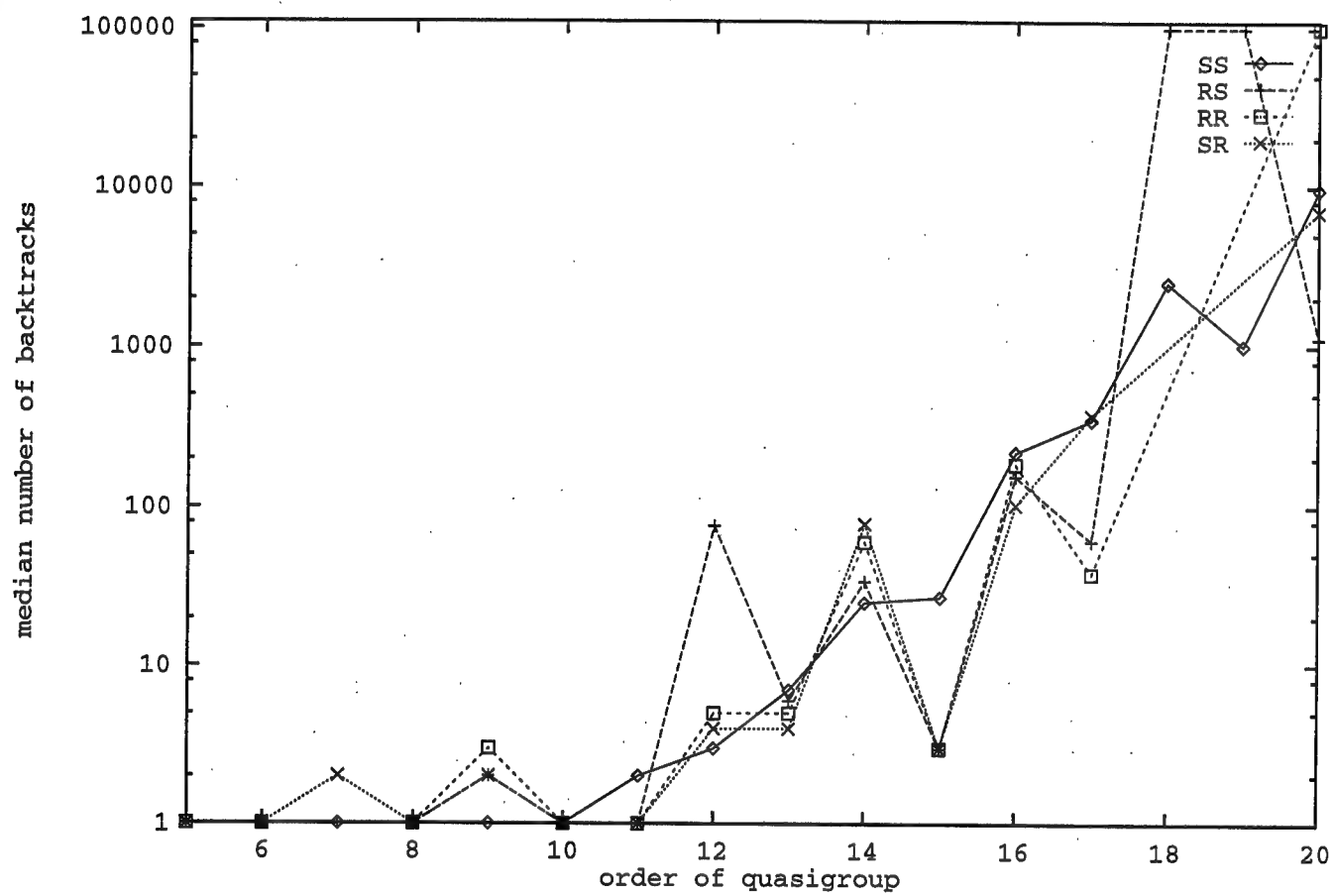Figure 10: Finding quasigroups with 5% pre-assigned values.

Figure 11: Finding quasigroups at the phase transition.

6 Distribution Profiles

Our stochastic First-Fail strategies are examples of Las Vegas style algorithms. They always return a correct answer (either showing consistency of proving inconsistency), but the running time of the procedures varies from run to run due to the random element.

Figure 12 shows the distribution of the "RR" procedure for finding a quasigroup of order 20 (no pre-assigned values).[1] The distribution is obtained over 500 runs of "RR" on the quasigroup problem. The deterministic First-Fail procedure takes 645 backtracks on this instance. From the figure, we see that "RR" takes less than 30 backtracks in 75% of the runs. This suggests that the best way of running this kind of procedure is either in parallel or sequentially using several short runs (e.g., a cutoff of 100).

Nevertheless, care should be taken in the use of stochastic procedures, because they can have a remarkably wide range of running times. For example, "RR" also has some chance of making bad choices. In fact, it takes 645 or more than 645 backtracks on about 12% of the runs.[2] In general, short runs with rapid restarts appears to be the best way to exploit the random element of such procedures.

In their recent Science paper, Huberman *et al.* (1997) discuss the importance of considering the variance of the performance distribution of an algorithm as a measure of the quality of the algorithm. In fact, they view it as a measure of the inherent risk associated with an algorithm, in the sense that it determines how much variation its performance can have. Drawing an analogy with a typical approach in economics, where utility is optimized considering a given level of risk, they suggest the construction of "portfolios" of algorithms with a whole range of performance and risk characteristics. By using such an approach, one can optimize the use of computational resources considering the desired level of performance and risk. In our case, a portfolio of algorithms could be obtained, for example, by combining runs of "RR" with runs of "RS". "RS" has an expected run time value somewhat higher than "RR", but it has a much smaller variance.

Finally, Figure 13 gives the distribution profile of "RR" on an *inconsistent* quasigroup completion problem. The deterministic First-Fail procedure takes

---

[1]The other two stochastic procedure have similar distribution profiles.

[2]"SR" took only a maximum of 400 backtracks on our sample of 500 runs.

2946 backtracks to show inconsistency of this instance. We see how "RR" outperforms the deterministic strategy. In fact, on this instance, "RR" never took more than 1373 backtracks in our sample of 500 runs. Also, the mode of the distribution lies around 400. Again, given the run time profile, one can design an optimal strategy for running the procedure either in parallel or sequentially, using short runs.

The main point we would like to stress here is that even though we have to cover the entire search space to prove the non-existence of the completion of the quasigroup, the stochastic branching and variable selection leads to significant reduction of the expected overall size of the search tree (needed to show inconsistency). So, although the usefulness of randomization is a well-established fact for local procedures, our results suggest that a random element is also a useful addition to a complete, systematic method, even for discovering inconsistencies.

We have provided experimental results showing the potential payoff of adding a stochastic element to complete search methods. The payoff is largest on the most structured problem instances. Our results show that stochasticity can also be useful in showing *inconsistency*. This complements the well-known success of using randomness in incomplete model-find procedures.

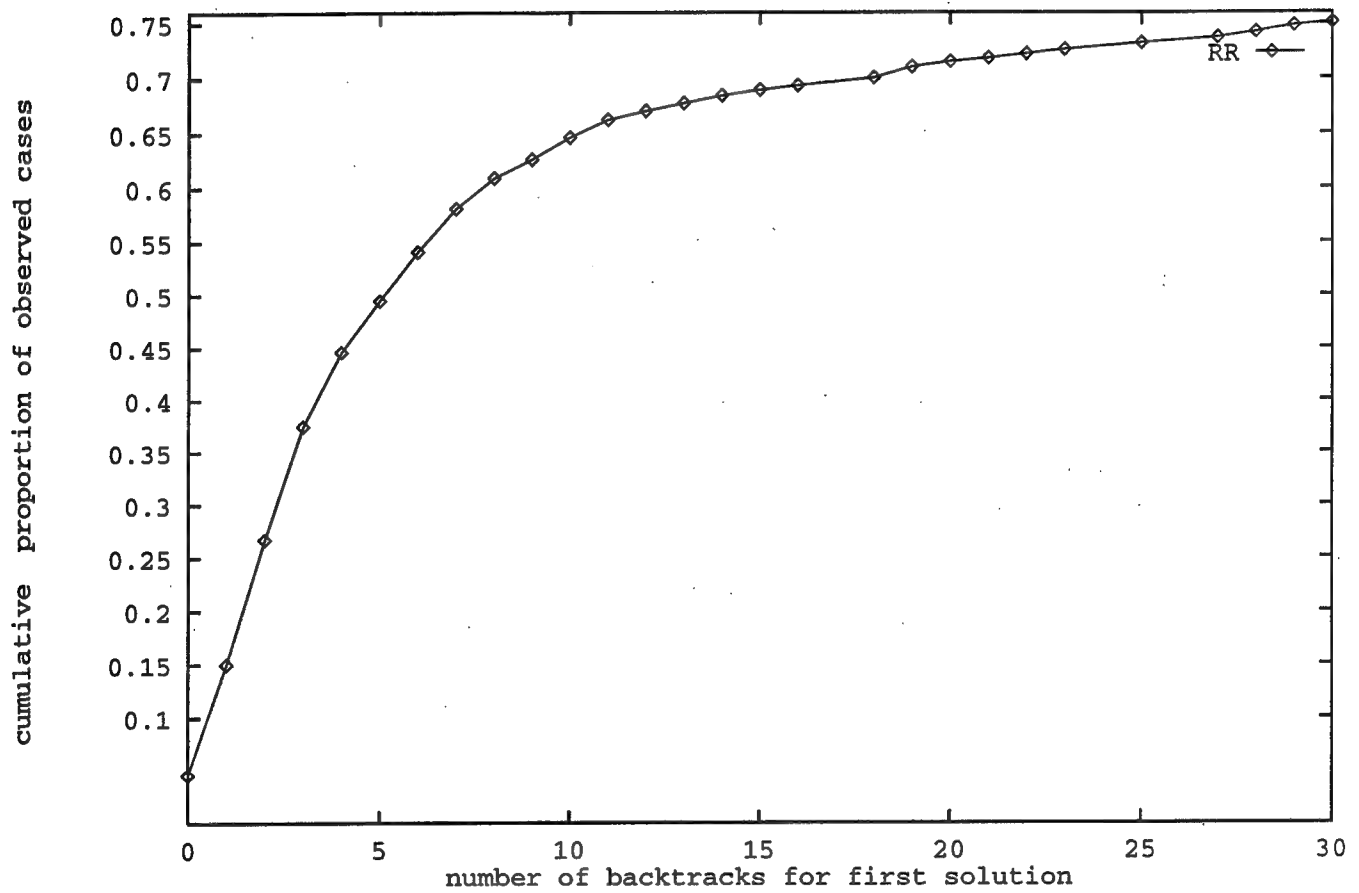In the next section we study the design of *portfolios* of stochastic algorithms.

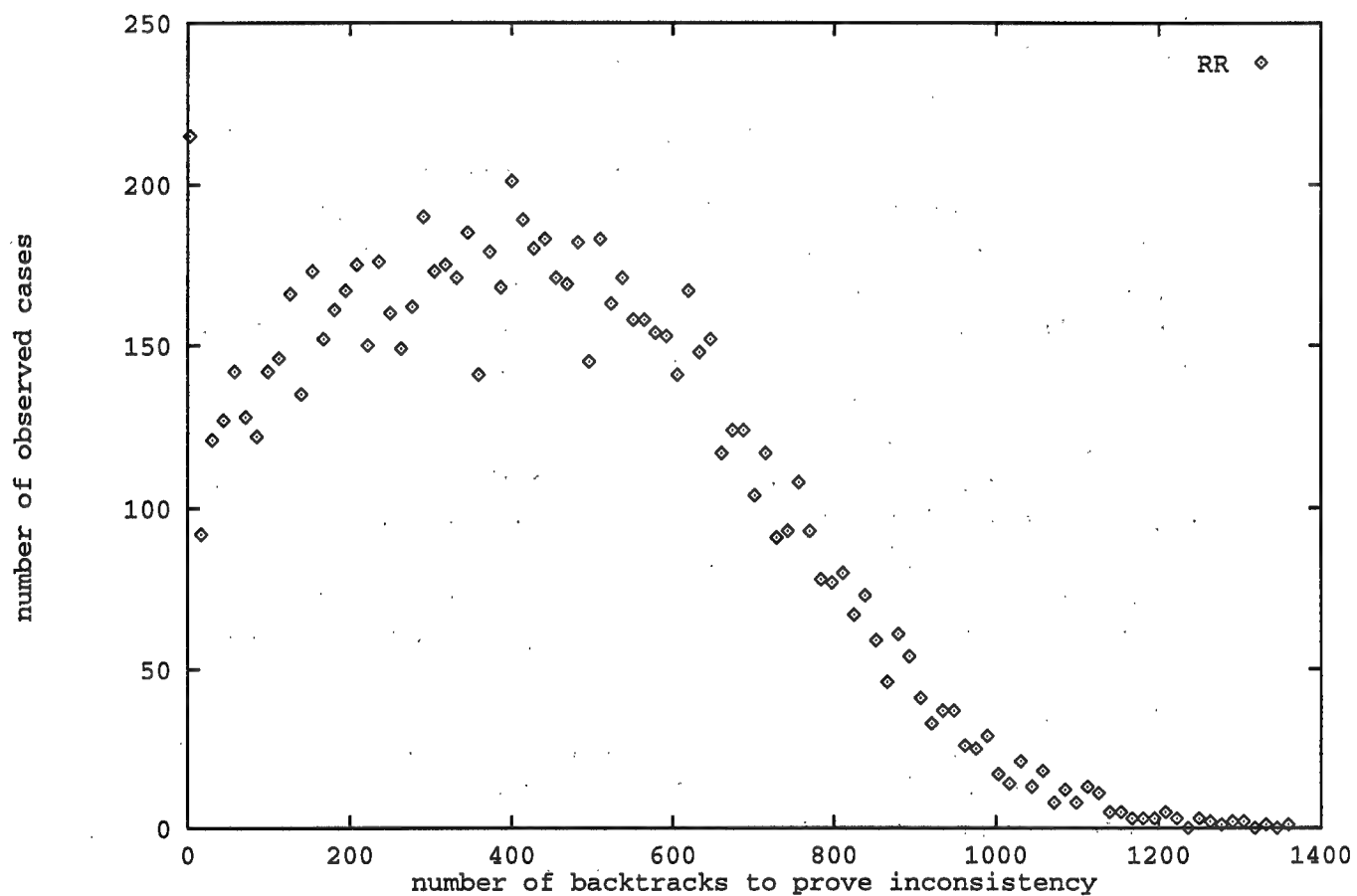Figure 12: Distribution for *finding* quasigroups of order 20 (no pre-assignment).

Figure 13: Distribution for *proving* inconsistency. (Order 10 with 43% pre-assignment).

7 Portfolio Design

A *portfolio of algorithms* is merely a collection of different algorithms and/or different copies of the same algorithm running on different processors.[3] Here we consider the case of independent runs without interprocess communication. We are considering Las Vegas type algorithms, *i.e.*, stochastic algorithms that always return a model satisfying the constraints of the search problem or demonstrate that no such model exists (Motwani and Raghavan 1995). The computational cost of the portfolio is therefore a random variable. The *expected* computational cost of the portfolio is simply the expected value of the random variable associated with the portfolio and its *standard deviation* is a measure of the "dispersion" of the computational cost obtained when using the portfolio of algorithms. In this sense, the standard deviation is a measure of the risk inherent to the portfolio.

The main motivation to combine different algorithms into a portfolio is to improve on the performance of the component algorithms, mainly in terms of expected computational cost, but also in terms of the overall *risk*. As we will show, some portfolios are strictly preferable to others, in the sense that they provide a lower risk and also a lower expected computational cost. However, in some cases, we cannot identify within a set of portfolios the one that is best both in terms of expected value and risk. This set of portfolios corresponds to the *efficient set* or *efficient frontier*, following terminology used in the theory of mathematical finance. Within this set of portfolios, in order to minimize the risk, one has to deteriorate the expected value or, in order to improve the expected value of the portfolio, one has to increase the risk.

In this context, where we characterize a portfolio in terms of its expected value and variance, combining different algorithms into a portfolio only makes sense if they exhibit different probability profiles and none of them *dominates* the others over the whole spectrum of problem instances. Algorithm A dominates algorithm B if the cumulative frequency distribution of algorithm A lies above the cumulative frequency distribution of algorithm B for all points.[4]

Let us consider a set of two algorithms, *algorithm* 1 and *algorithm* 2.

---

[3] One can also consider the somewhat more general case of interleaving the execution of algorithms on one or more processors.

[4] Another criterion for combining algorithms into a portfolio is given by the *correlation* of the algorithms.

We assume that neither one strictly dominates the other. Let us associate a random variable with each algorithm:

A1 - the number of backtracks that algorithm 1 takes to find the first solution or to prove that a solution does not exist

A2 - the number of backtracks that algorithm 2 takes to find the first solution or to prove that a solution does not exist

Let us assume that we have N processors and that we design a portfolio using $n1$ processors with *algorithm* 1 and $n2$ processors with *algorithm* 2. So, $N = n1 + n2$. Let us define the random variable associated with this portfolio:

X - the number of backtracks that the portfolio takes to find the first solution or to prove that a solution does not exist

The probability distribution of X is a "weighted" probability distribution of the probability distributions of *algorithm* 1 and *algorithm* 2. More precisely, the probability that $X = x$ is given by the probability that one processor takes exactly x backtracks and all the other ones take x or more backtracks to find a solution or to prove that a solution does not exist.

Let us assume that we have N processors and our portfolio consists of N copies of *algorithm* 1. In this case, P[X=x] is given by the probability that one processor take exactly x backtracks and the other $(N - 1)$ take more than x backtracks, plus the probability that two processors take exactly x backtracks and the other $(N - 2)$ ones take more than x backtracks, etc., plus the probability that all the processors take exactly x backtracks to find a solution or to prove that a solution does not exist. The following expression gives the probability function for such portfolio.

Given N processors, and let $n1 = N$ and $n2 = 0$. P[X=x] is given by

$$\sum_{i=1}^{N} \binom{N}{i} P[A1 = x]^i P[A1 > x]^{(N-i)}$$

If we now consider two algorithms we just have to generalize the above expression, considering that $X = x$ can occur just within the processors that use *algorithm* 1, or just within the processors that use *algorithm* 2, or

33

within both. As a result, the probability function for a portfolio with two algorithms is given by the following expression:

Given N processors, $n1$ and $n2$ such that $0 \leq n1 \leq N$, and $0 \leq n2 \leq N$. P[X=x] is given by

$$\sum_{i=1}^{N}\sum_{i'=0}^{n1} \binom{n1}{i'} P[A1 = x]^{i'} P[A1 > x]^{(n1-i')} \times \binom{n2}{i''} P[A2 = x]^{i''} P[A2 > x]^{(n2-i'')}]$$

The value of i'' is given by $i'' = i - i'$, and the term in the summation is 0 whenever $i'' < 0$ or $i'' > n2$.

In the case of a portfolio involving two algorithms the probability distribution of the portfolio is a summation of a product of two expressions, each one corresponding to one algorithm. In the case of a portfolio comprising M different algorithms, this probability function can be easily generalized, by having a summation of a product of M expressions, each corresponding to a algorithm.

Having derived the probability distribution for the random variable associated with the portfolio, the calculation of its *expected value* and *standard deviation* is straightforward.

7.1 Empirical results for portfolio design

We consider a popular extension of the First-Fail heuristic, called the Brelaz heuristic (Brelaz 1979). The Brelaz heuristic was originally introduced for graph coloring procedures. It provides one of the most powerful graph-coloring and general CSP heuristics (Trick and Johnson 1996).

The Brelaz heuristic specifies a way for breaking ties in the First-Fail rule: If two variables have equally small remaining domains, the Brelaz heuristic proposes to select the variable that shares constraints with the largest number of the remaining unassigned variables. A natural variation on this tie-breaking rule is what we call the "reverse Brelaz" heuristic, in which preference is given to the variable that shares constraints with the *smallest* number of unassigned variables. Any remaining ties after the (reverse) Brelaz rule are resolved randomly. One final issue left to specify in our search procedure is the order in which the values are assigned to a variable. In the standard Brelaz, value assignment is done in lexicographical order (*i.e.*, systematic). In our experiments, we consider four strategies:

- *Brelaz-S* — Brelaz with systematic value selection,

- *Brelaz-R* — Brelaz with random value selection,

- *R-Brelaz-S* — Reverse Brelaz with systematic value selection, and

- *R-Brelaz-R* — Reverse Brelaz with random value selection.

Figure 14 shows the performance profile of our four strategies for the problem of finding a quasigroup of order 20 (no pre-assigned values). Each curve gives the cumulative distribution obtained for each strategy by solving the problem 10,000 times. The cost (horizontal axis) is measured in number of backtracks, which is directly proportional to the total runtime of our strategies. For example, the figure shows that R-Brelaz-R finished roughly 80% of the 10,000 runs in 15 backtracks or less. The top panel of the figure shows the overall profile; the bottom part gives the initial part of the profile.

First, we note that that R-Brelaz-R dominates R-Brelaz-S over the full profile. In other words, the cumulative relative frequency curve for R-Brelaz-R lies above that of R-Brelaz-S at every point along the x-axis. R-Brelaz-S, in turn, strictly dominates Brelaz-R. As we will see below, we often encounter such patterns, where one strategy simply consistently outperforms other strategies. Unfortunately, this leaves no room for combining strategies: one simply picks the best strategy. This may explain why some of the ideas about combining algorithms has not received much attention in the traditional communities that deal with hard computational problems.[5]

From the perspective of combining algorithms, what is most interesting, however, is that in the initial part of the profile (see bottom panel, Figure 14), Brelaz-S dominates R-Brelaz-R. Intuitively, Brelaz-S is better than R-Brelaz-R at finding solutions *quickly*. However, in the latter part of the cumulative distribution (for more than five backtracks), R-Brelaz-R dominates Brelaz-S. In a sense, R-Brelaz-R gets relatively better when the search gets harder. As we will see in the next section, we can exploit this in our algorithm portfolio design.

Figure 15, shows the performance profiles for quasigroups with 10% pre-assigned values. We see essentially the same pattern as in Figure 14, but the region where Brelaz-S dominates is relatively smaller.

---

[5]There is still the issue of multiple runs with the same method. We'll return to this below.
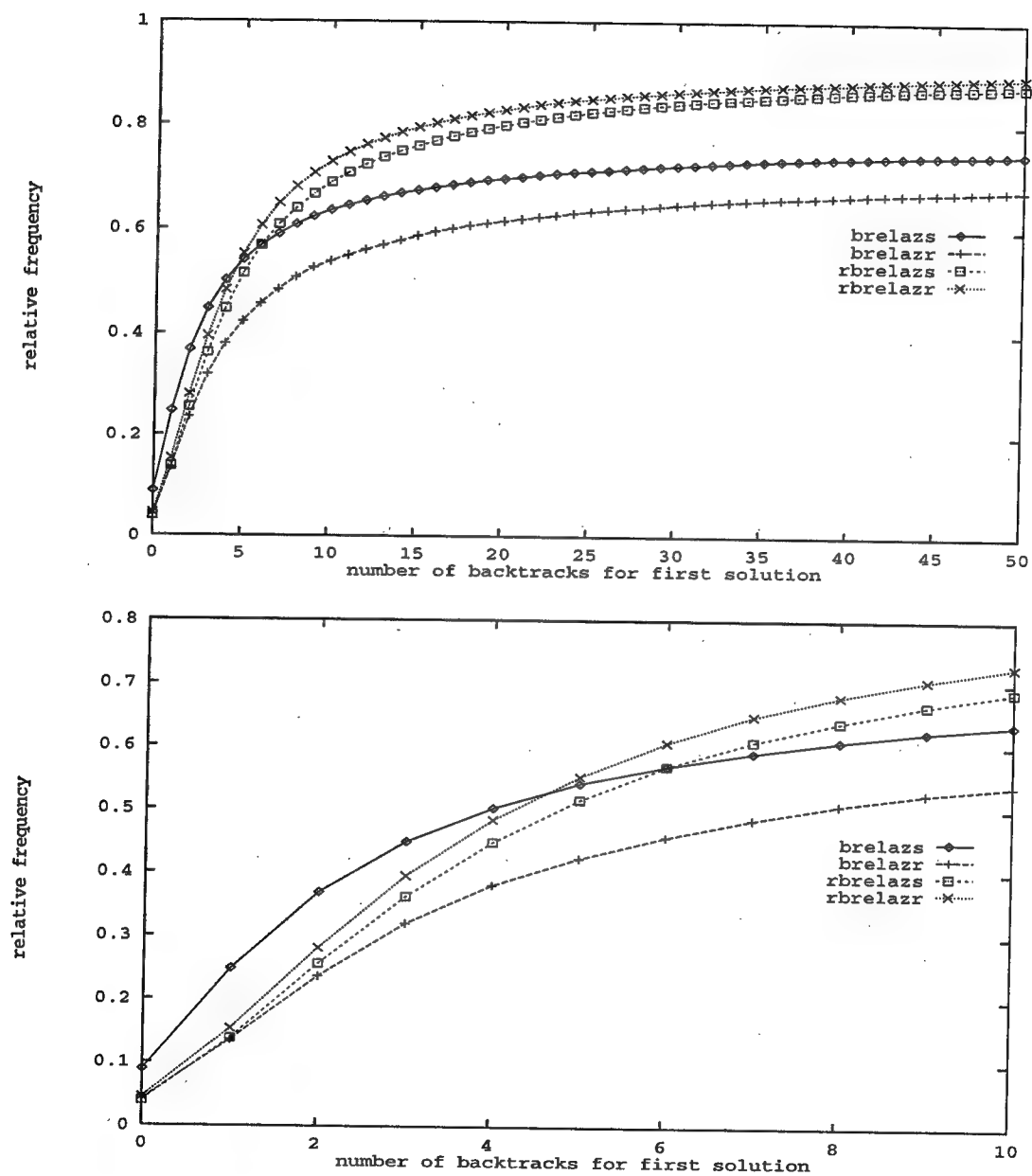
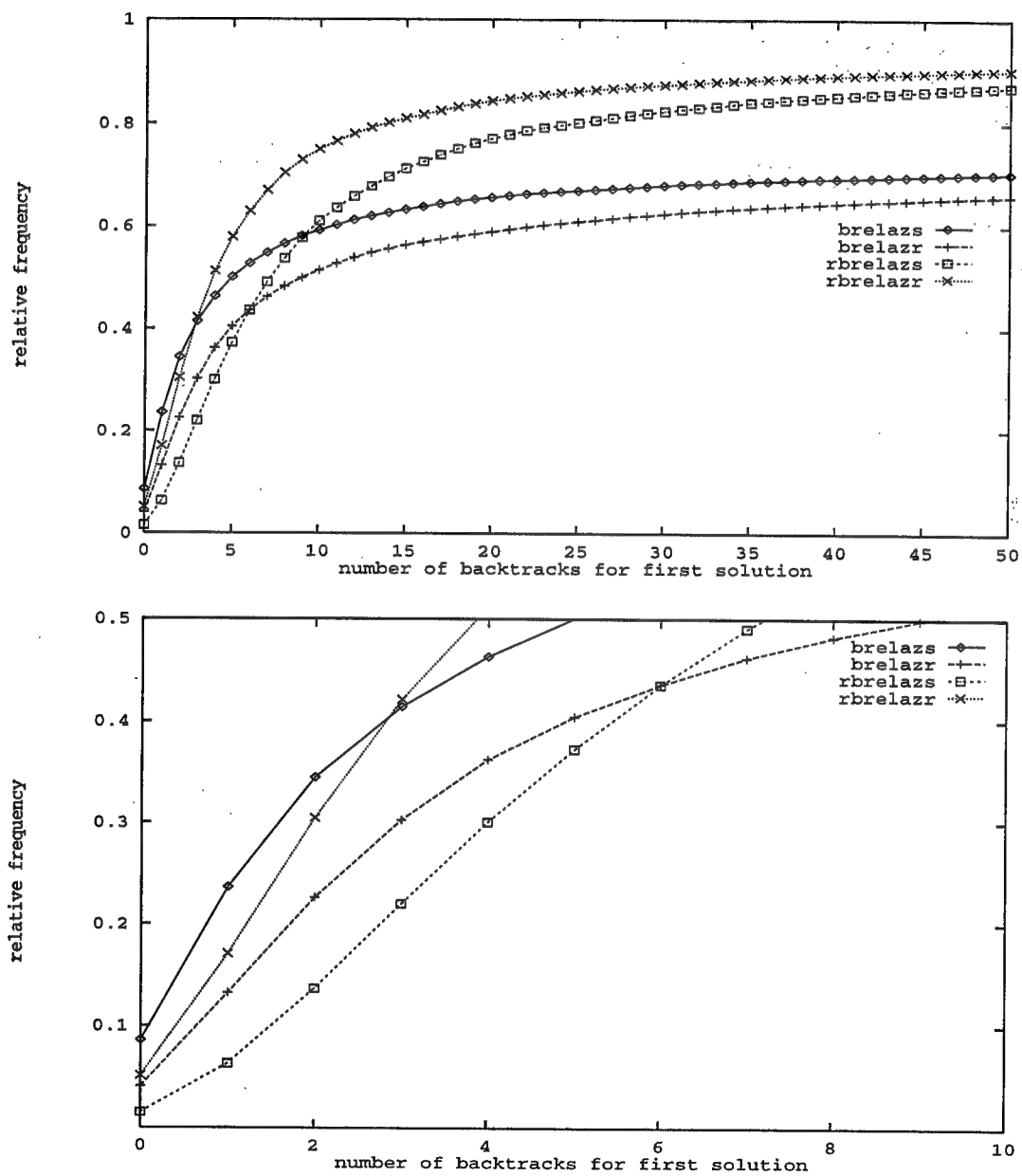Figure 14: Finding quasigroups of order 20 (no pre-assigned values).

Figure 15: Finding quasigroups of order 20 with 10% pre-assigned values.
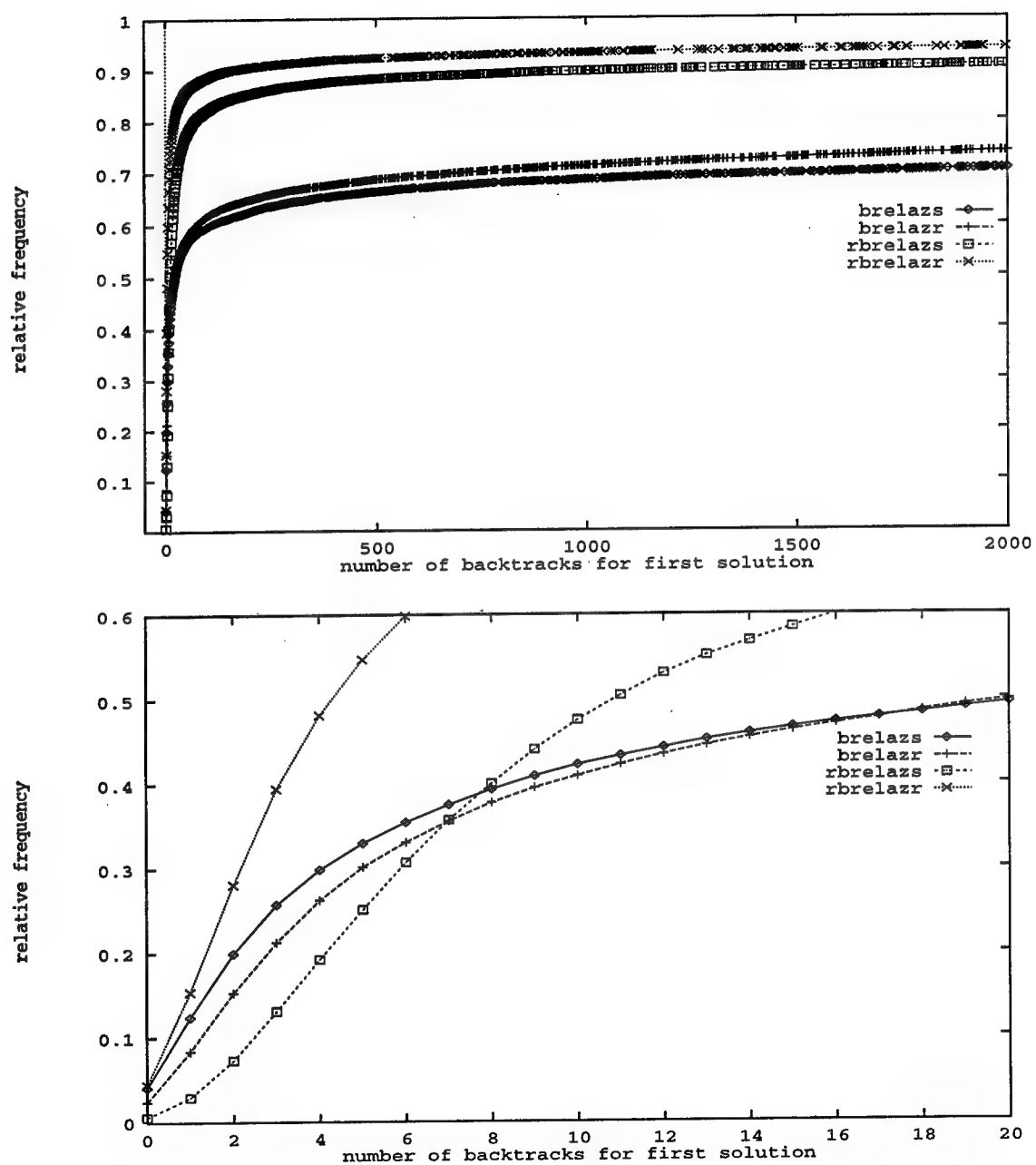
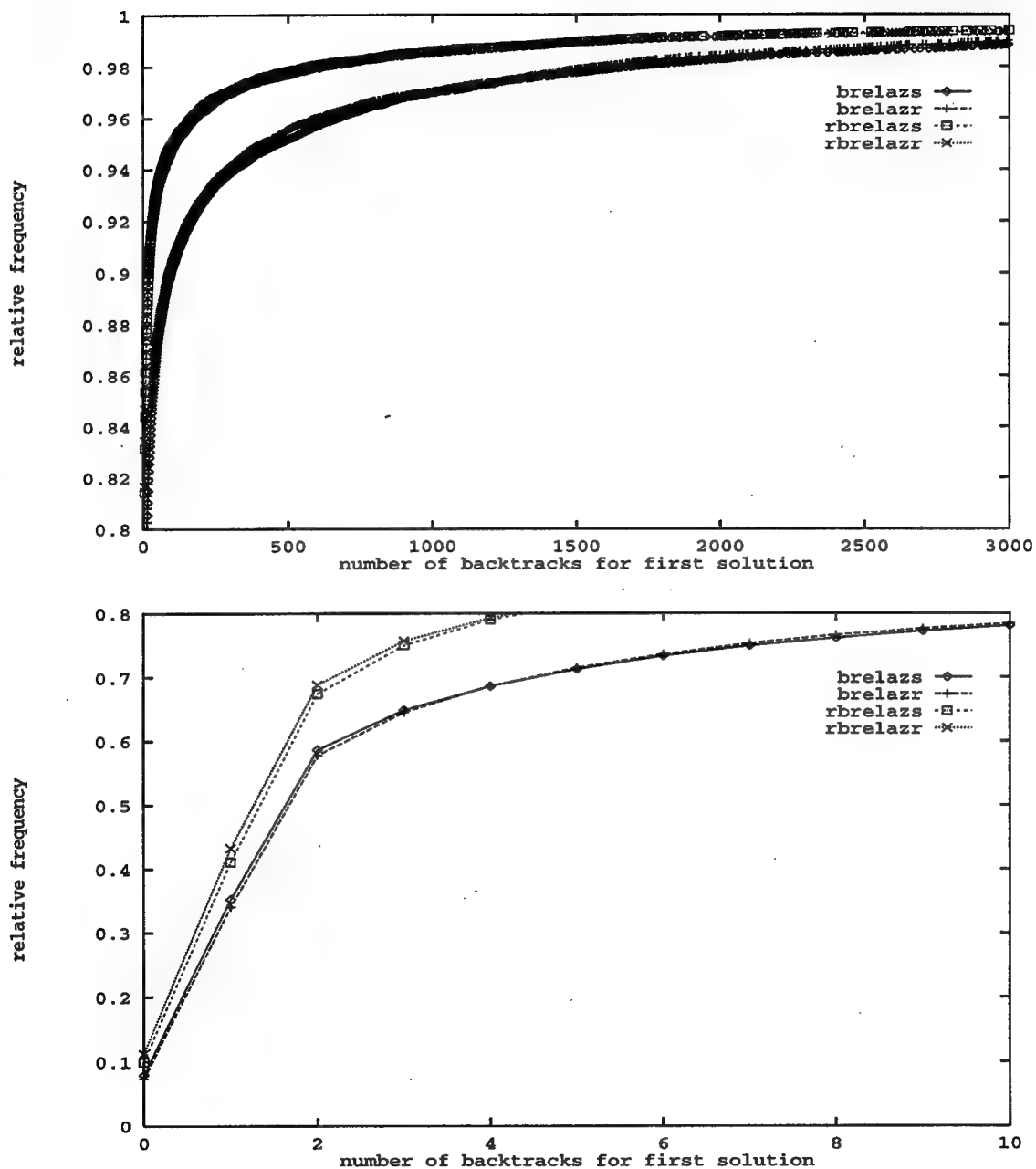Figure 16: Finding quasigroups of order 20 with 20% pre-assigned values.

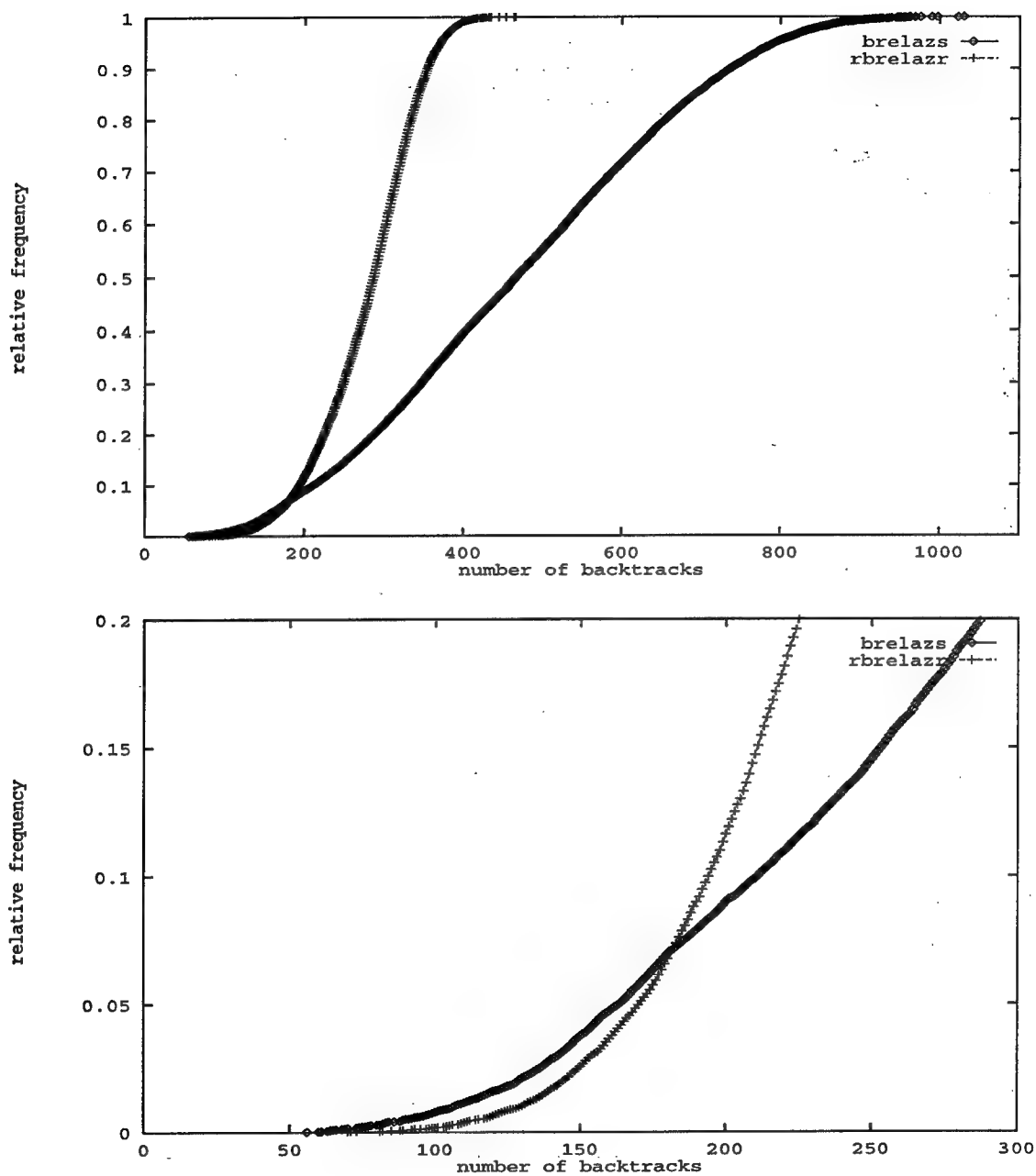Figure 17: Finding quasigroups of order 20 at the phase transition.

Figure 18: Showing inconsistency of completion of quasigroup (order 10 with 43% pre-assigned values).

When we increase the percentage of pre-assigned values (20% pre-assigned, Figure 16), we see that R-Brelaz-R completely dominates the other strategies over the whole problem spectrum. This pattern continues for the higher numbers of pre-assigned values (Figure 17, at the phase transition with roughly 40% pre-assigned).

Finally, Figure 18 gives the performance profile for showing the inconsistency of a quasigroup completion problem. The instance in question has 43% pre-assigned values. Here we again observe that Brelaz-S is somewhat better at finding inconsistencies quickly but again R-Brelaz-R dominates for most of the profile. Again, the good initial performance of Brelaz-S can be exploited by combining many short runs, as we will see below.

We now design different portfolios based on our performance profiles. We focus on the case of finding a quasigroup of order 20 with no-preassigned values. The performance profiles are given in Figure 14. Note that this is an interesting case from the portfolio design perspective because Brelaz-S dominates in the initial part of the distribution, whereas R-Brelaz-R dominates in the latter part.

Figures 19, 20, 21, and 22 give the expected values and the standard deviations of portfolios for 2, 5, 10, and 20 processors, respectively. (Results derived using the formula given above.) We see that for 2 processors (Figure 19), the portfolio consisting of two copies of R-Brelaz-R has the best expected value *and* the lowest standard deviation. This portfolio dominates the two other 2-processor portfolios.

When we increase the number of processors, we observe an interesting shift in the optimal portfolio mix. For example, for 5 processors, using 2 Brelaz-S gives a better expected value at only a slight increase in the risk (standard deviation) compared to zero Brelaz-S. In this case, the efficient set comprises three portfolios. One with 5 R-Brelaz-R, one with 1 Brelaz-S and 4 R-Brelaz-R, and one with 2 Brelaz-S and 3 R-Brelaz-R. The situation changes even more dramatically if we go to yet more processors. In particular, with 20 processors (Figure 22), the best portfolio corresponds to using all processors to run the Brelaz-S strategy (the lowest expected value and the lowest standard deviation). The intuitive explanation for this is that by running many copies of Brelaz-S, we have a good chance that at least one of them will find a solution quickly. This result is consistent with the common

use of "random restarts" in stochastic search methods in practical applications. It's interesting to see that our rigorous analysis confirms this intuition. Our portfolio analysis also gives the somewhat counter-intuitive result that, even when given two stochastic algorithms, where neither strictly dominates the other, running multiple copies of a single algorithm is preferable to a mix of algorithms (under certain resource constraints; Figure 19 and Figure 22).

We have provided concrete empirical results showing the computational advantage of a portfolio approach for dealing with hard combinatorial search and reasoning problems as compared to the best more traditional single algorithm methods. Our analysis also showed what properties of the problem instance distributions lead to the largest payoff for using a portfolio approach in practice. Finally, we saw how the use of random restarts of a good stochastic method is often the optimal strategy. These results suggest that ideas developed in the flexible computation community can play a significant role in practical algorithm design.
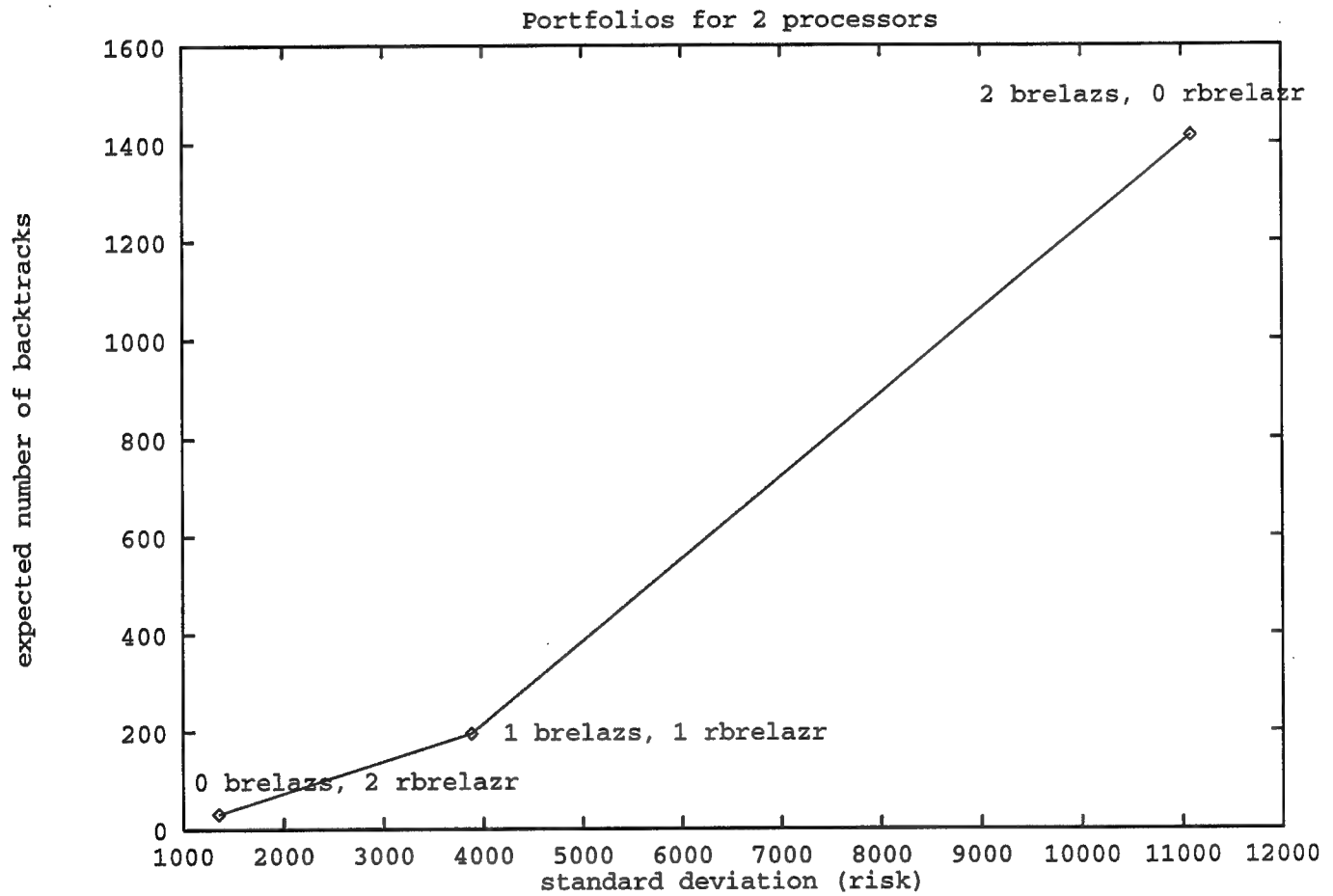
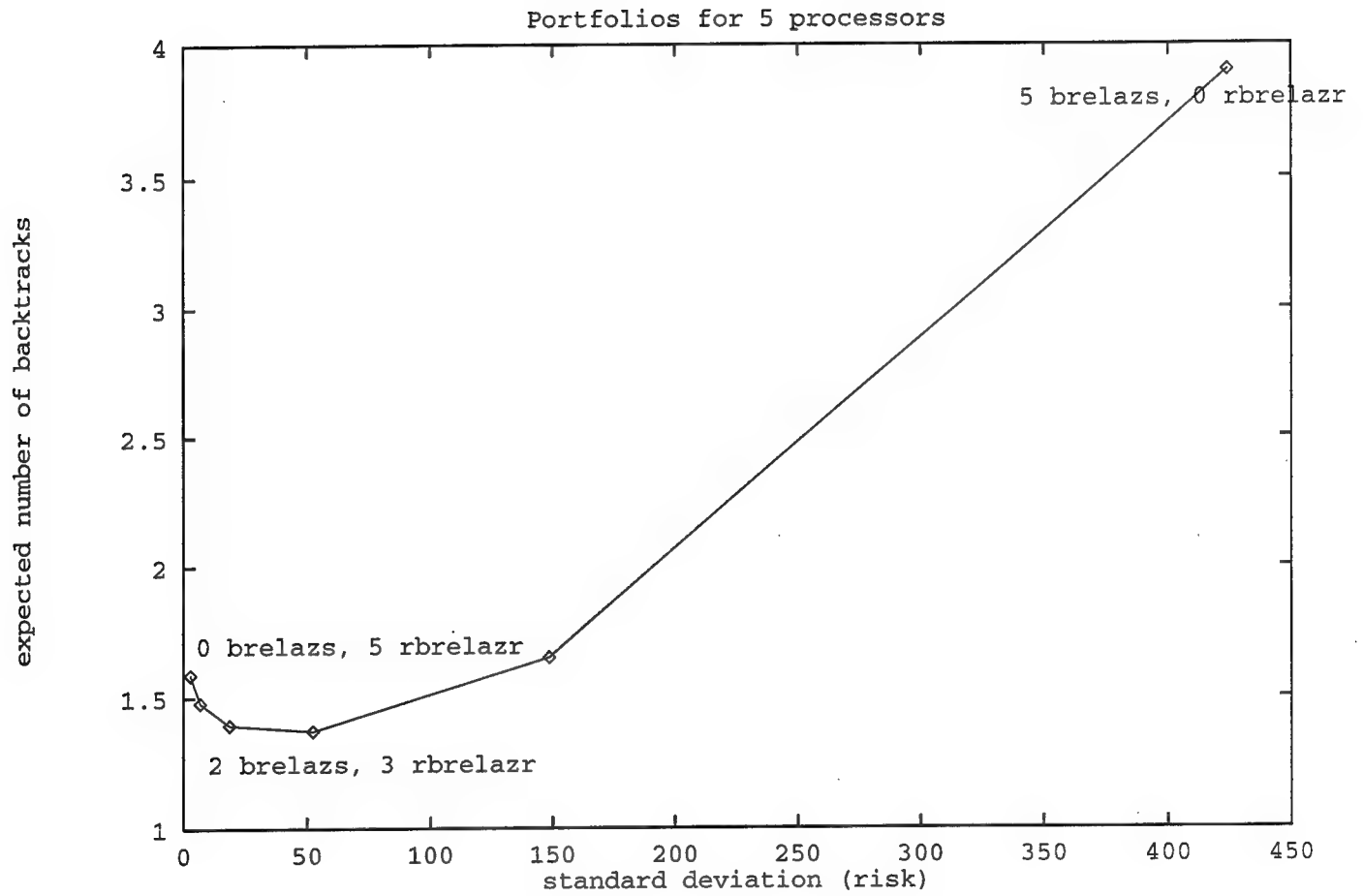Figure 19: Portfolios for two processors combining Brelaz and R-Brelaz-R.

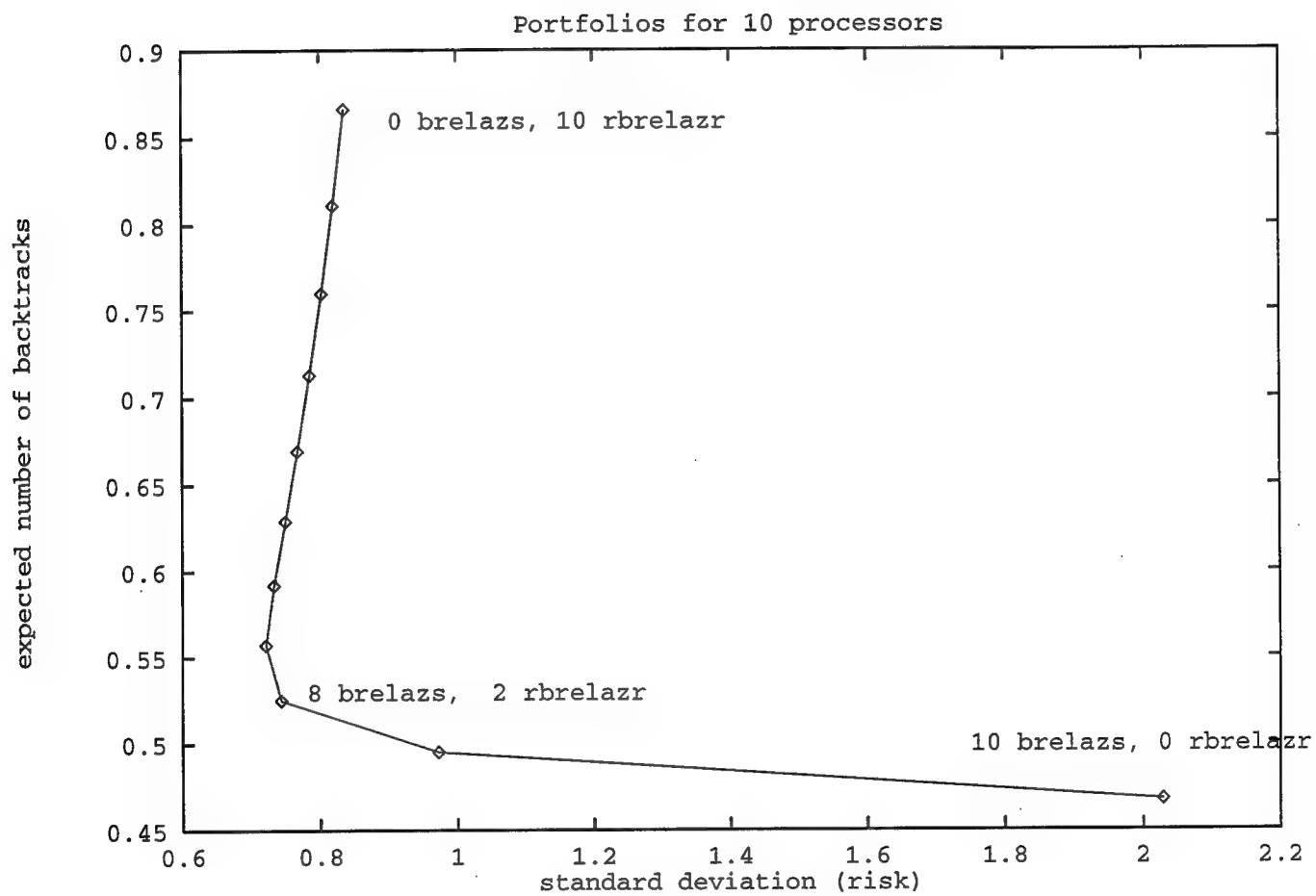Figure 20: Portfolios for five processors combining Brelaz and R-Brelaz-R.

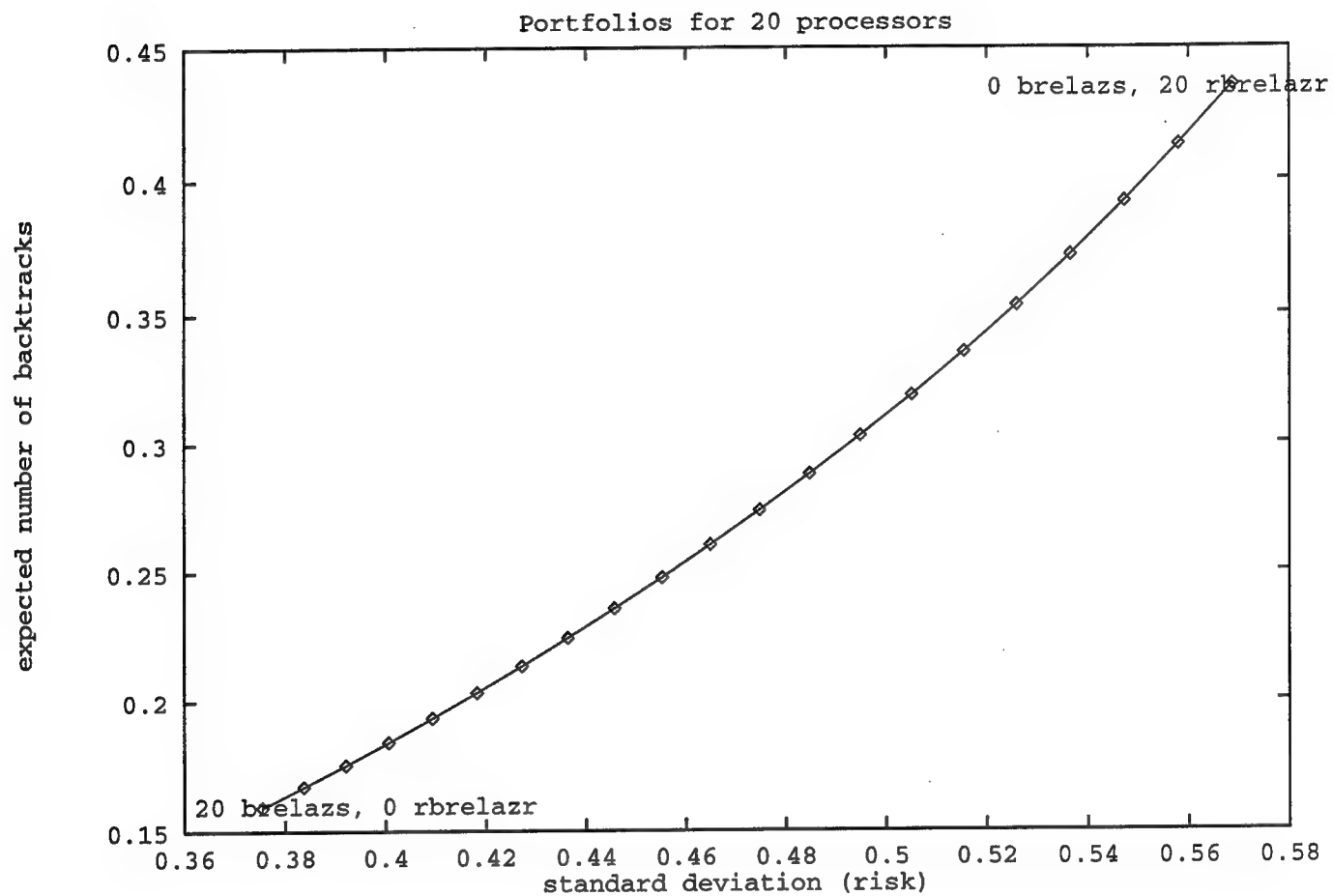Figure 21: Portfolios for ten processors combining Brelaz and R-Brelaz-R.

Figure 22: Portfolios for twenty processors combining Brelaz and R-Brelaz-R.

8 Conclusions

We propose a new benchmark problem for the evaluation of search procedures, the quasigroup completion problem. This problem is very challenging from a computational point of view. Furthermore, this domain offers a tremendous research potential for the study of different aspects of search methods applied to structured domains. In fact, not only does the quasigroup domain possess inherent interesting structural properties, we can easily add additional structural properties to this domain. As an example, we might require that the quasigroup be idempotent.

Using the quasigroup completion problem as a benchmark, we showed that it can be very counterproductive to design a search procedure that mimics a particular constructive method. As a matter fact, a specialized search control, designed to mimic a constructive method for the quasigroups, degrades rapidly in the presence of even small perturbations. On the other hand, a more general heuristic is more robust in the presence of perturbations.

The realization that small perturbations improved the performance of a general heuristic led us to study the effect of adding a stochastic element directly into a deterministic complete search method. Our results on the quasigroup completion problem showed that stochastic complete search methods can improve the performance of a deterministic complete search method dramatically. Nevertheless, care should be taken, since these results are measured probabilisticly.

We pushed the idea of exploiting stochasticity one step further by combining algorithms that exhibit competing behavior in terms of expected performance, measured in terms of the expected number of backtracks of the algorithm, and in terms of risk, measured in terms of the variance of the algorithm, into a *portfolio* of algorithms. Our results showed that a portfolio approach can have a dramatic impact in terms of the overall performance, in comparison to the performance of each of its component algorithms. Moreover, our results clearly show that the optimal mix of algorithms is directly dependent on the resources available. An interesting special case is when the best strategy consists of combining copies of the same algorithm. This *portfolio* is analogous to the practice of "restarts" for stochastic procedures, where the same algorithm is run repeatedly with different seeds, a common practice in the theorem-proving community. Combining copies of the same algorithm into a *portfolio* might be a good strategy, especially in the cases

that one algorithm *dominates* for a given class of problems.

Acknowledgments

References

Andersen, L. (1985). Completing Partial Latin Squares. *Mathematisk Fysiske Meddelelser*, 41, 1985, 23–69.

Andersen, L. and Hilton, J. (1983). Thank Evans! *Proc. London Math. Soc.*, (3) 47 (1983), 507-522.

Brelaz, D. (1979). New methods to color the vertices of a graph. *Comm. of the ACM* (1979) 251–256.

Bruen A. and Dixon R. (1974). The N-Queens Problem. *Discrete Mathematics*, vol. 12, 1975, 393–395.

Cheeseman, Peter and Kanefsky, Bob and Taylor, William M. (1991). Where the Really Hard Problems Are. *Proceedings IJCAI-91,* 1991, 163–169.

Chvatal,V. and Szemeredi, E. (1988). Many hard examples for resolution. *JACM*, val. 35, no. 4, 1988, 759–208.

Colbourn, C. (1983). Embedding Partial Steiner Triple Systems is NP-Complete. *J. Combin. Theory* (A) 35 (1983), 100-105.

Colbourn, C. (1984). The Complexity of Completing Latin Squares. *Discrete Appl. Math.*, 8, (1984), 25-30.

Crawford, J. and Auton, L. (1996) Experimental Results on the Crossover Point in Random 3-SAT. *Artificial Intelligence*, 81, 31–57.

Dean, T. and Boddy, M. (1988) An analysis of time-dependent planning. *Proc. AAAI-88*, St. Paul, MI (1988) 49–54.

Dechter, R. (1991) Constraint networks. *Encyclopedia of Artificial Intelligence* John Wiley, New York (1991) 276-285.

Denes, J. and Keedwell, A. (1974) Latin Squares and their Applications. *Akademiai Kiado, Budapest, and English Universities Press*, London, 1974.

Erbas C., Tanik, M., and Aliyazicioglu, Z. (1992) Linear Congruence Equations for the solutions of the N-Queens Problem. *Information Processing Letters*, 41 (April 1992) 301-306.

Evans, T. (1960) Embedding Incomplete Latin Squares. *Amer. Math.*, 67 (1960), 958–961.

Falkowski, B. and Schmitz, L. (1986). A Note on the N-Queens' Problem. *Information Processing Letters*, 23, 1986, 39–46.

Freuder, F., Dechter, R., Ginsberg, M., Selman, B., and Tsang, E. Systematic versus stochastic constraint satisfaction. *Proc. IJCAI-95*, Montreal, Canada (1995).

Freuder, E. and Mackworth, A. (Eds.). *Constraint-based reasoning*. MIT Press, Cambridge, MA, USA, 1994.

Fujita, M., Slaney, J., and Bennett, F. (1993). Automatic Generation of Some Results in Finite Algebra *Proc. IJCAI*, 1993.

Gent, I.P. , MacIntyre, E., Prosser, Smith, B., and Walsh, T. (1996) An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem. *Proc. CP-96*, Boston, MA, 179–193.

Gent, I. and Walsh, T. (1995) The TSP Phase Transition. First Intl. Workshop on AI&OR. Timberline, OR, 1995.

Gent, I. and Walsh, T. (1996) The Satisfiability Constraint Gap. *Artificial Intelligence*, 81, 1996.

Ginsberg, M. and Geddis, D. (1991) Is there a need for domain-dependent control information? *Proc. AAAI-91*.

Gomes, C. and Selman, B. (1997a). Problem structure in the presence of perturbations. To appear in *Proc. AAAI-97*.

Gomes, C. and Selman, B (1997b). Algorithm Portfolio Design - Theory vs. Practice. To appear in *Proc. UAI-97*.

Horvitz, E. and Zilberstein S. (1996) (Eds.) *Proceedings of Flexible Computation*, AAAI Fall Symposium, Cambridge, MA, 1996.

Huberman, B.A., Lukose, R.M., and Hogg, T. (1997). An economics approach to hard computational problems. *Science*, 265, 51–54.

Hogg, T., Huberman, B.A., and Williams, C.P. (Eds.) (1996). Phase Transitions and Complexity. *Artificial Intelligence*, 81 (Spec. Issue; 1996)

Hoffman, E. J., Loessi, J.C., and Moore, R. C. (1969). Constructions for the N-Queens Problem. *Math. Mag.*, March-April (1969), 66–72.

Kirkpatrick, S. and Selman, B. (1994) Critical Behavior in the Satisfiability of Random Boolean Expressions. *Science*, 264 (May 1994) 1297–1301.

Kondrak, G and van Beek, P. (1995) A theoretical evaluation of selected backtracking algorithms. *Proc. IJCAI-95*, 1995, 541–547.

Lam, C., Thiel, L., and Swiercz, S. (1989) The Non-existence of Finite Projective Planes of Order 10. *Can. J. Math.*, Vol. XLI, 6, 1989, 1117-1123.

Mitchell, D., Selman, B., and Levesque, H.J. (1989) Hard and easy distributions of SAT problems. *Proc. AAAI-92*, San Jose, CA (1992) 459–465.

McCune, M. (1996) Solution of the Robbins Problem. Draft, 1996.

Minton, S. (1996) Automatically configuring constraint satisfaction programs: A case study. *Constraints*, 1 (1).

Motwani, R. and Raghavan, P. (1995) *Randomized algorithms.* Cambridge, 1995.

Puget, J.-F. (1994) A C++ Implementation of CLP. *Technical Report 94-01* ILOG S.A., Gentilly, France, (1994).

Russell, S and Norvig P. (1995) *Artificial Intelligence a Modern Approach.* Prentice Hall, Englewood Cliffs, NJ. (1995).

Schiex, T. and Verfaillie, G. Nogood Recording for Static and Dynamic Constraint Satisfaction Problems. *Proc. Int. Conf. on Tools with AI*, 1993.

Smetaniuk, B. (1981) A New Construction on Latin Squares - 1: A Proof of the Evans Conjecture. *Ars Combinatoria* XI (1981), 155-172.

Smith, B. and Dyer, M. Locating the Phase Transition in Binary Constraint Satisfaction Problems. *Artificial Intelligence*, 81, 1996.

Trick, M. and Johnson, D. (Eds.) (1996) *Proc. DIMACS Challenge on Satisfiability Testing, Graph Coloring, and Cliques.* DIMACS Series on Discr. Math., Am. Math. Soc. Press (1996).

Tsang, E.P.K. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

van Hentenryck, P. , Deville, Y., and Teng Choh-Man (1992) A generic arc consistency algorithm and its specializations. *Artificial Intelligence*, 57, 1992.

Williams, C.P. and Hogg, T. (1992) Using deep structure to locate hard problems. *Proc. AAAI-92,* San Jose, CA, July 1992, 472–277.

Yaglom, A. M. and Yaglom, i. M. (1964) *Challenging Mathematical Problems with Elementary Solutions.* Holden-Day, San Francisco, CA, 1964, 92-98.

Zhang, W. and Korf, R. A Study of Complexity Transitions on the Asymmetric Traveling Salesman Problem. *Artificial Intelligence,* 81, 1996.

MISSION
OF
ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Material Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.